Neurodynamische Module zur Bewegungssteuerung autonomer mobiler Roboter

DISSERTATION

zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften (doctor rerum naturalium)
im Fach Informatik

eingereicht an der
Mathematisch-Naturwissenschaftlichen Fakultät II
Humboldt-Universität zu Berlin

von
Herrn Diplom-Mathematiker Manfred Hild
geboren am 4. April 1968 in Konstanz, Deutschland

Präsident der Humboldt-Universität zu Berlin:
Prof. Dr. Christoph Markschies

Dekan der Mathematisch-Naturwissenschaftlichen Fakultät II:
Prof. Dr. Wolfgang Coy

Gutachter:

1. Prof. Dr. Hans-Dieter Burkhard, Humboldt-Universität zu Berlin, Institut für Informatik
2. Prof. Dr. Frank Pasemann, Universität Osnabrück, Institut für Kognitionswissenschaften
3. Prof. Dr. Ralf Der, Universität Leipzig, Institut für Informatik

eingereicht am:	16. April 2007
Tag der mündlichen Prüfung:	21. September 2007

Abstract

How recurrent neural networks can help to make autonomous robots move, will be investigated within this thesis. First, oscillators which are able to control four-legged robots will be dealt with, then homeostatic ring modules which control segmented robots, and finally monostable neural modules, which are able to drive complex motion sequences on robots with many degrees of freedom will be focused upon.

The mathematical theory of neural modules will be addressed as well as their practical implementation on real robot platforms. This includes their embedding into a major framework and concrete aspects, like computational accuracy, timing and dependance on materials. Details on electronics will be given, so that individual robot systems can be built and equipped with an appropriate motion controller.

It is another concern of this thesis, to shed a new light on the theory of recurrent neural networks, from the perspective of classical engineering science. Selective comparisons to analog electronic schematics, physical models, and digital signal processing algorithms can ease the understanding of neural dynamics.

Keywords:
Recurrent Neural Networks, Neurodynamics, Neuromodule, Dynamical Systems, Learning Rule, Homeostasis, Autonomous Mobile Robots, Humanoid Robots, Robotics, Motion Control, Oscillator, Ring Oscillator, Sensorimotor Loop

Zusammenfassung

In der vorliegenden Arbeit werden rekurrente neuronale Netze im Hinblick auf ihre Eignung zur Bewegungssteuerung autonomer Roboter untersucht. Nacheinander werden Oszillatoren für Vierbeiner, homöostatische Ringmodule für segmentierte Roboter und monostabile Neuromodule für Roboter mit vielen Freiheitsgraden und komplexen Bewegungsabläufen besprochen.

Neben dem mathematisch-theoretischen Hintergrund der Neuromodule steht in gleichberechtigter Weise deren praktische Implementierung auf realen Robotersystemen. Hierzu wird die funktionale Einbettung ins Gesamtsystem ebenso betrachtet, wie die konkreten Aspekte der zugrundeliegenden Hardware: Rechengenauigkeit, zeitliche Auflösung, Einfluss verwendeter Materialien und dergleichen mehr. Interessante elektronische Schaltungsprinzipien werden detailliert besprochen. Insgesamt enthält die vorliegende Arbeit alle notwendigen theoretischen und praktischen Informationen, um individuelle Robotersysteme mit einer angemessenen Bewegungssteuerung zu versehen.

Ein weiteres Anliegen der Arbeit ist es, aus der Richtung der klassischen Ingenieurswissenschaften kommend, einen neuen Zugang zur Theorie rekurrenter neuronaler Netze zu schaffen. Gezielte Vergleiche der Neuromodule mit analogen elektronischen Schaltungen, physikalischen Modellen und Algorithmen aus der digitalen Signalverarbeitung können das Verständnis von Neurodynamiken erleichtern.

Schlagwörter:
Rekurrente Neuronale Netze, Neurodynamik, Neuromodul, Dynamische Systeme, Lernregel, Homöostase, Autonome Mobile Roboter, Humanoide Roboter, Robotik, Bewegungssteuerung, Oszillator, Ringoszillator, Sensomotorische Schleife

Vorwort

Diese Dissertationsschrift fasst die Ergebnisse von fünf Jahren Forschungs- und Entwicklungsarbeit zusammen. Ausgangspunkt war hierbei die Suche nach einer Systemarchitektur für autonome, mobile Roboter.

Im Laufe der Arbeiten an dem umfassenden Themengebiet, hat sich die Bewegungssteuerung als besonders spannendes und gleichzeitig anspruchsvolles Teilgebiet herauskristallisiert. Dies gilt insbesondere für humanoide Roboter, deren Entwicklung und Ansteuerung im Vergleich zur industriellen Robotik weltweit noch in den Kinderschuhen steckt.

Sprache und Notation

Die vorliegende Arbeit ist bewußt in Deutsch verfasst und trotzt damit dem allgemeinen Trend, alles ins Englische zu übersetzen. Abgesehen davon, dass man sich in seiner Muttersprache stets am besten ausdrücken kann, bietet die Deutsche Sprache einen sehr differenzierten Wortschatz, mit dessen Hilfe sich Sachverhalte präzise und nuanciert beschreiben lassen. Als Orientierung diente der Leitfaden [Sch01] von Wolf Schneider, dem ehemaligen Leiter der Hamburger Journalistenschule. Es wird durchgängig die in Deutschland übliche Notation mathematischer Symbole verwendet. Eingeführte Begriffe und Bezeichnungen werden über die Kapitel hinweg konsistent gehalten.

Inhaltsverzeichnis

Teil I

Theorie

Kapitel 1

Einleitung

Mit dem Begriff *Roboter* verbinden die meisten Leute hochautomatisierte Fertigungsstraßen, wie sie heute in jeder Autofabrik zu finden sind. Die wesentlichen Eigenschaften von Industrierobotern sind: Schnelligkeit, Positionier- und Wiederholgenauigkeit. Weitere wichtige Eigenschaften sind Kraft, exakte Programmierbarkeit der Verfahrwege und die Möglichkeit, das Werkzeug zu wechseln. Industrieroboter werden in der Regel stationär eingesetzt. Sie sind schwer, teuer, verbrauchen viel Energie und müssen für jede Aufgabe neu programmiert werden. Obwohl ihre Entwicklung vor Jahrzehnten begonnen hat, ist ein Ende nicht in Sicht. Entsprechende Publikationen beschäftigen sich mit der Steigerung von Effizienz und Präzision dieser Systeme (siehe [BS03]).

Den Kontrapunkt zu Industrierobotern bilden die autonomen, mobilen Roboter, denn diese benötigen völlig andere Fähigkeiten. Aufgrund ihrer Mobilität müssen sie die Energiequelle mitführen, daher sollten sie eher leicht gebaut sein und wenig Energie verbrauchen. Bewegt sich ein mobiler Roboter von A nach B, dann kommt es nicht so sehr darauf an, dass er einen exakten Weg einhält, sondern vielmehr, dass er beispielsweise mit unvorhersehbaren Unebenheiten des Bodens zurechtkommt. Fällt er einmal um, sollte er einerseits genügend propriozeptive Sensoren besitzen um dies festzustellen – andererseits aber auch die motorische Fähigkeit besitzen, wieder aufstehen zu können.

Fragestellung und Motivation

Anfang der 90er Jahre prägten Raymond Reiter und Erik Sandewall den Begriff *Cognitive Robotics* (Kognitive Robotik). Er bezeichnet das Gebiet der Künstlichen Intelligenz, welches sich mit geeigneten Wahrnehmungsmechanismen für autonome, mobile Roboter auseinandersetzt. Der Titel „Neuro-

dynamische Module zur Bewegungssteuerung autonomer mobiler Roboter“ ist insofern wie folgt in die zentrale Fragestellung der vorliegenden Arbeit zu überführen: *Wie lässt sich eine Bewegungssteuerung mit Hilfe neuronaler Netze so realisieren, dass später die notwendigen Sensorsignale eingekoppelt werden können?*

Bewegungssteuerung kann im allgemeinen nicht als isolierte, rein motorische Fragestellung behandelt werden. Vielmehr ist es oft erst die sensomotorische Interaktion mit der Umwelt, die einem Roboter sinnvolle Bewegungen erlaubt. Ein ausführliches Beispiel hierzu findet man in [NF00]. Roboter haben sehr unterschiedliche Morphologien und Freiheitsgrade. Aus diesem Grund lautet die nächste Frage: *Welche unterschiedlichen Mechanismen der Bewegungssteuerung lassen sich gegeneinander abgrenzen?* Ziel ist hierbei nicht eine erschöpfende Darstellung aller denkbaren Mechanismen, sondern eine exemplarische Abdeckung der wesentlichen Modi.

Die Hauptmotivation dieser Arbeit besteht darin, den Entwicklern von autonomen, mobilen Robotersystemen neue, theoretisch untermauerte und praktisch erprobte Lösungsansätze anzubieten. Ein weiteres Anliegen ist es, von den klassischen Ingenieurswissenschaften ausgehend, einen neuen Zugang zur Theorie neuronaler Netze anzubieten.

Die inhaltliche Einbettung dieser Arbeit in einen globalen wissenschaftlichen Kontext geht über den Vergleich mit bereits etablierten Lösungen hinaus. Es wird gezeigt, wie derartige Lösungen in die Theorie neuronaler Netze übersetzt werden können und umgekehrt, wie gewisse Neuromodule mit klassischen Methoden implementiert werden können.

Aufbau der Arbeit

Die vorliegende Dissertationsschrift besteht aus einem theoretischen und einem praktischen Teil, die sich über viele Querverweise gegenseitig ergänzen. Dennoch ist jeder der beiden Teile inhaltlich in sich geschlossen und problemlos ohne den jeweils anderen zu verstehen, so dass man, je nach Neigung, auch getrost zunächst in den praktischen Teil einsteigen kann.

Der theoretische Teil beschreibt für die Bewegungssteuerung relevante neuronale Netze. Es empfiehlt sich, mit der Lektüre von Kapitel 2 zu beginnen, da dort die Definition des zugrundeliegenden Neuronenmodells gegeben wird. In Kapitel 3 werden wichtige Eigenschaften rekurrenter neuronaler Netze erläutert. Auf diese Eigenschaften wird im gesamten theoretischen Teil immer wieder verwiesen.

In Kapitel 4 werden unterschiedliche Oszillatoren betrachtet, mit denen sich vierbeinige autonome Roboter ansteuern lassen. Das darauf folgende Kapitel 5 erweitert die vorgestellten Strukturen. Es wird eine homöostatische

Lernregel hergeleitet und ein Neuromodul vorgestellt, mit dem sich die Bewegungen segmentierter Roboter steuern lassen.

Kapitel 6 handelt von der Steuerung komplexer Bewegungssequenzen bei Robotern mit vielen Freiheitsgraden, zum Beispiel humanoiden Robotern. Es wird beschrieben, wie existierende Bibliotheken von Bewegungsdaten in ein neuronales Netz übersetzt werden können und welche Chancen daraus erwachsen. Kapitel 7 fasst alle Resultate zusammen und schließt damit den theoretischen Teil ab.

Im praktischen Teil werden die gebauten Robotersysteme vorgestellt. Kapitel 8 liefert eine chronologische Übersicht aller Systeme und zeigt auf, inwiefern Konzepte weiterentwickelt und wiederverwendet wurden. In den drei folgenden Kapiteln werden der Reihe nach *stationäre Systeme*, *radgetriebene Roboter* und *Laufmaschinen* beschrieben. Die Kapitel sind inhaltlich weitestgehend voneinander unabhängig.

Kapitel 12 fasst die Ergebnisse des praktischen Teils zusammen und stellt die wesentlichen Kernaussagen nochmals explizit dar. Die Arbeit schließt mit einem Ausblick auf noch offene Fragen und weiterführende Arbeitspläne.

Methodik

In dieser Arbeit werden neuronale Netze vorgestellt, die der Bewegungssteuerung von Robotern dienen. Jedes Netz wird hierbei dem folgenden methodischen Kanon unterzogen:

- Definition des Zielverhaltens
- Etablierung eines geeigneten Netzes
- Simulation und Analyse des Netzes
- Praktische Erprobung auf einem realen System

Die Definition des Zielverhaltens und die praktische Erprobung auf einem realen Robotersystem bilden die methodische Klammer und sind im wesentlichen selbsterklärend. Besonderes Augenmerk verdient hingegen die Etablierung der neuronalen Netze, bei welcher drei unterschiedliche Methoden zum Einsatz kamen:

- Künstliche Evolution
- Transfer nicht-neuronaler Strukturen
- Komposition komplexer Netze aus einfachen Neuromodulen

Von diesen drei Methoden ist die künstliche Evolution die unvoreingenommenste Vorgehensweise. Man definiert ein Zielverhalten und macht zunächst keine weiteren Annahmen über die mögliche Lösungsstruktur. Dies ist etwas vereinfacht gesprochen, da selbstverständlich bereits die Verwendung neuronaler Netze als implizite Annahme verstanden werden kann. Jedoch erfordert die künstliche Evolution im Vergleich mit den anderen Methoden keine schöpferische Kreativität.

Der Transfer nicht-neuronaler Strukturen bietet sich in den Bereichen an, wo bereits Standardlösungen in den klassischen Ingenieurswissenschaften existieren. Hiervon wird bei den Oszillatoren in Kapitel 4 ausführlich Gebrauch gemacht. Alternativ lassen sich auch physiologische Erkenntnisse aus der Neurobiologie adaptieren. Diese liegen in der Regel bereits in Form einer neuronalen Struktur vor, wenngleich ihnen ein anderes Neuronenmodell zugrunde liegt, als das hier verwendete.

Schließlich bietet sich als dritte und anspruchsvollste Methode die theoretische Herleitung beziehungsweise gezielte Komposition komplexer neuronaler Netze an. Beispielsweise die Netze zur Steuerung komplexer Bewegungssequenzen in Kapitel 6 gehören in diese Kategorie.

Die Methodik, mit der ein bestimmtes Netz generiert wurde, bestimmt zugleich, welche Analysen sinnvollerweise durchgeführt werden können. Hat man das Netz aus verschiedenen, funktional in sich geschlossenen Neuromodulen zusammengesetzt, so macht es wenig Sinn die Attraktoren des Gesamtsystems zu betrachten. Anders sieht der Fall bei Netzen aus, die per künstlicher Evolution gefunden wurden. Hier kann das Studium der Attraktoren wichtige Hinweise auf die Funktionsweise liefern.

Neben der deskriptiven Analyse ist es häufig sinnvoll, zusätzlich mathematische Größen zu betrachten, wie zum Beispiel den größten Lyapunov-Exponenten einer Neurodynamik sowie die Eigenwerte der entsprechenden Verbindungsmatrix. Darüber hinaus liefert die numerische Simulation oftmals den besten Einblick in die Funktionsweise. Die mathematische Analyse der Attraktoren kann dann nämlich auf die Bereiche eingeschränkt werden, in denen das Netz beim Betrieb auf einem realen Robotersystem auch arbeitet. Nicht immer wird die komplette Dynamik eines Netzes ausgenutzt.

Weitere Aspekte ergeben sich bei der Implementierung eines neuronalen Netzes auf einem realen Robotersystem. Der Einfluss von Rechengenauigkeit, zeitlicher Auflösung und vielen anderen Parametern ist nicht zu vernachlässigen und daher ebenfalls Gegenstand der durchgeführten Analysen.

Bemerkung zur künstlichen Evolution

Auf die künstliche Evolution wird in dieser Arbeit nicht gesondert eingegangen. Es wird das in der evolutionären Robotik übliche Standardverfahren eingesetzt, wie in [NF00] beschrieben. Gearbeitet wurde mit einer Populationsgröße von $N = 20$. Die besten $N_s = 8$ Individuen wurden in die nächste Generation übernommen (*Selektion*) und der Rest der Population mit per *Crossover* generierten Individuen aufgefüllt. Auf die gesamte Population wurde eine *Mutation* angewandt, deren Rate dynamisch so gewählt wurde, dass die durchschnittliche Fitness bei 75 Prozent der besten Fitness zu liegen kam, wie in Abbildung 1.1 links zu sehen ist.

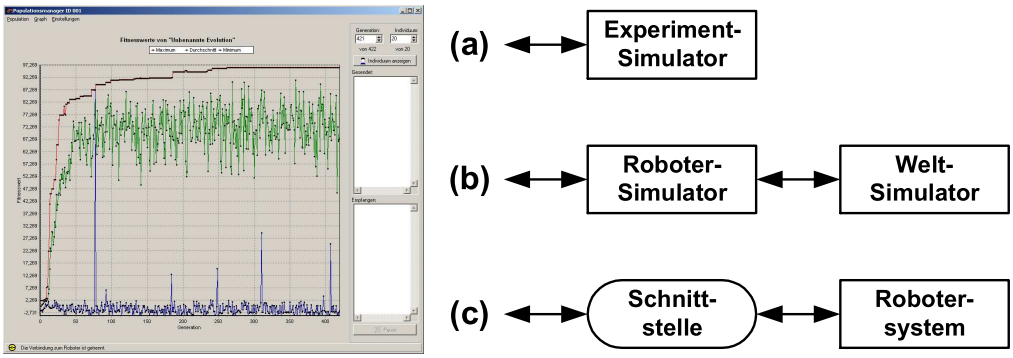

Abbildung 1.1: *(links) Der Populationsmanager ist ein Programm, das die künstliche Evolution neuronaler Netze durchführt. Die oberste Kurve gibt den zeitlichen Verlauf der besten Fitnesswerte an, darunter sind die Kurven der durchschnittlichen und schlechtesten Fitnesswerte zu sehen. (rechts) Am Populationsmanager können unterschiedliche Programme betrieben werden, wie zum Beispiel die Simulation eines bestimmten Experiments (a), eines Roboters in einer Umwelt (b), oder eine Schnittstelle zu einem existierenden Robotersystem (c), so dass auf echter Hardware evolviert werden kann.*

Der abgebildete Populationsmanager sowie der Weltsimulator wurden im Rahmen von Seminararbeiten programmiert und ausgiebig getestet (siehe [KB03] und [Urb03]). Neben reinen Simulationen wurden auch Experimente mit real gebauten Robotersystemen durchgeführt, beispielsweise die Evolution energetisch günstiger Bewegungsverläufe beim Universalgreifer (siehe Bemerkung zur sensomotorischen Schleife in Kapitel 9.2). Auch die mit einem Kunstkopf durchgeführten Experimente zur Phonotaxis zählen hierzu (siehe Kapitel 9.4 und [Bau05]).

Anstelle einer Anbindung des Populationsmanagers an Hardware, lässt sich auch ein vereinfachter Evolutionsalgorithmus direkt im Mikroprozessor des Robotersystems unterbringen. Man spricht dann von einer sogenannten

On-Board-Evolution. Im Rahmen einer Studienarbeit wurde dies erfolgreich von Ferry Bachmann durchgeführt. Der notwendige Versuchsaufbau ist in Abbildung 1.2 zu sehen.

Die in Kapitel 10.1 näher beschriebene Fahrende Platine dient als Plattform. Sie wird von einem neuronalen Netz gesteuert, dessen Verbindungsgewichte zu Beginn zufällig gewählt sind, weshalb die Fahrende Platine ständig aneckt. Im Laufe des Experiments werden die Gewichte von der Evolution so verändert, dass die Fahrende Platine Hindernissen ausweicht. Der vollständige Evolutionsverlauf dauert mehrere Stunden. Alle Resultate sowie die Details der Vorgehensweise sind in [Bac05] zu finden.

Abbildung 1.2: *Versuchsaufbau zur On-Board-Evolution von Hindernisvermeidung mit der Fahrenden Platine. Der quadratische Experimentalbereich hat eine* $1.44m^2$ *große Grundfläche. Über das graue Kabel erfolgt die Stromzufuhr.*

Kapitel 2

Neuronale Netze

Im Jahr 1943 veröffentlichten der Neurobiologe McCulloch und der Statistiker Pitts das Modell eines Neurons, welches als universelle Grundlage mathematischer Berechnungen dienen kann [MP43].

Ihr biologisch angelehntes, einfaches Modell inspirierte bis heute die Entwicklung etlicher Derivate, die sich grob anhand von drei Kriterien kategorisieren lassen:

1. Kontinuierliches/diskretes Zeitmodell
2. Amplituden-/frequenzkodierte Signale
3. Feed-Forward-/rekurrente Vernetzung

Die ersten beiden Kriterien lassen sich an einem einzelnen Neuron direkt unterscheiden, das dritte bezieht sich hingegen auf die komplette Verbindungsstruktur eines Neuronenverbandes.

Zeitkontinuierliche Modelle werden überwiegend bei der Simulation neurophysiologischer Sachverhalte angewandt, wo es auf Vergleichbarkeit der Resultate mit echt abgeleiteten Zellpotenzialen ankommt. Da die kontinuierlichen Modelle numerisch aufwendige Integrationsprozesse enthalten, sind sie für die Robotik nur bedingt interessant – vor allem, wenn die Rechenleistung von einem mikroprozessorgesteuerten autonomen Roboter erbracht werden muss. Das in dieser Arbeit verwendete Modell ist daher zeitdiskret.

Wie weiter unten beschrieben wird, ist ein Neuron stets mehr oder weniger stark aktiviert und überträgt diese Aktivierung an andere Neuronen. In biologischen Neuronen geschieht diese Übertragung frequenzkodiert: je höher die Aktivierung, desto mehr Impulse pro Zeiteinheit werden übermittelt. Eine sehr ausführliche Beschreibung dieser als *Spiking Neurons* bekannten Modelle findet man in [GK02].

Mit Hilfe spezieller Hardware lässt sich diese Neuronenart sehr effizient implementieren, entweder auf digitale Weise (meist in FPGAs, siehe hierzu beispielsweise [MT00] und [MRTAE00]), oder vollständig in analoger Schaltungstechnik (siehe [GCA] und [FGA00]). Ein diskret aufgebautes, elektronisches Beispielneuron mit Schaltplan findet man auch in Kapitel 9.1 auf Seite 85.

Die Verwendung amplitudenkodierter Signale ist mathematisch einfacher beschreibar, was sich nicht nur beim analysieren gegebener neuronaler Strukturen bemerkbar macht, sondern auch bei deren Implementierung auf Mikroprozessoren. Verwendet man digitale Signalprozessoren (DSPs), wie bei den Experimenten zur Phonotaxis (Kapitel 9.4), oder moderne Mikroprozessoren mit integrierter DSP-Funktionalität, wie bei den Experimenten zur Bewegungswahrnehmung (Kapitel 9.3), dann lässt sich die Implementierung sogar noch effizienter gestalten.

Aus den genannten Gründen werden in dieser Arbeit die zeitdiskreten, amplitudenkodierten Modelle eingehend betrachtet, sowohl mit Feed-Forward-, als auch mit rekurrenter Vernetzung. Wie das Modell im einzelnen aussieht und auf welche Art Neuronen vernetzt werden können, davon handeln die nun folgenden Kapitel.

2.1 Definition des Modells

Ein neuronales Netz besteht aus einer Menge von Neuronen, die miteinander verbunden sind und von denen ausgezeichnete Neuronen als Eingangsbeziehungsweise Ausgangsneuronen fungieren. Das von McCulloch und Pitts [MP43] vorgeschlagene Modell eines Neurons ist in Abbildung 2.1 zu sehen.

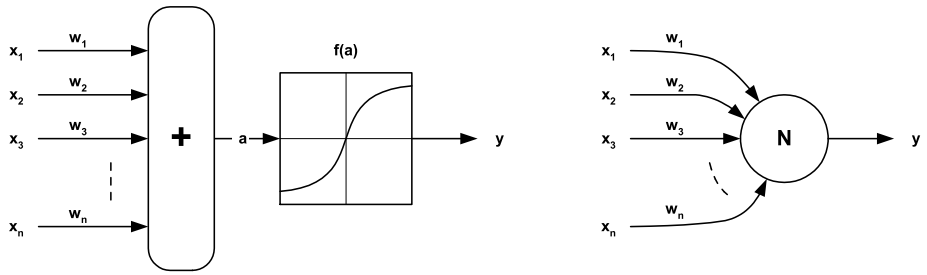

Abbildung 2.1: *(links) Neuronenmodell nach McCulloch und Pitts. (rechts) Vereinfachte Darstellung des Neurons.*

Es besitzt n Eingänge $x_1, x_2, \ldots, x_n$ und einen Ausgang y. Die Eingangssignale werden zunächst zu einer internen Neuronenaktivität a gewichtet aufaddiert. Danach wird das Augangssignal y über eine noch zu definierende

Abbildung f aus der Aktivität a gewonnen.

$$y := f(a), \qquad a := \sum_{i=1}^{n} w_i x_i, \qquad w_i, x_i \in \mathbb{R} \quad \text{für} \quad i = 1, 2, \ldots, n. \tag{2.1}$$

Die Gewichte w_i und Eingangssignale x_i fasst man zweckmäßigerweise jeweils zu einem n-dimensionalen Vektor zusammen. Da das Neuron nur zu diskreten Zeitschritten $t \in \mathbb{N}$ betrachtet wird, erhält man insgesamt:

$$\begin{aligned} y(t+1) &:= f(a(t+1)), \\ a(t+1) &:= \langle \boldsymbol{w}(t), \boldsymbol{x}(t) \rangle, \qquad \boldsymbol{w}(t), \boldsymbol{x}(t) \in \mathbb{R}^n, \ t \in \mathbb{N}. \end{aligned} \tag{2.2}$$

Sind die beiden Vektoren $\boldsymbol{w}$ und $\boldsymbol{x}$ betragsmäßig normiert, dann lässt sich ihr Skalarprodukt gemäß

$$\langle \boldsymbol{w}, \boldsymbol{x} \rangle = \|\boldsymbol{w}\| \cdot \|\boldsymbol{x}\| \cdot cos(\angle(\boldsymbol{w}, \boldsymbol{x})) \tag{2.3}$$

geometrisch als die Projektion des Eingangssignalvektors $\boldsymbol{x}$ auf den Gewichtsvektor $\boldsymbol{w}$ interpretieren, wie in Abbildung 2.2 dargestellt.

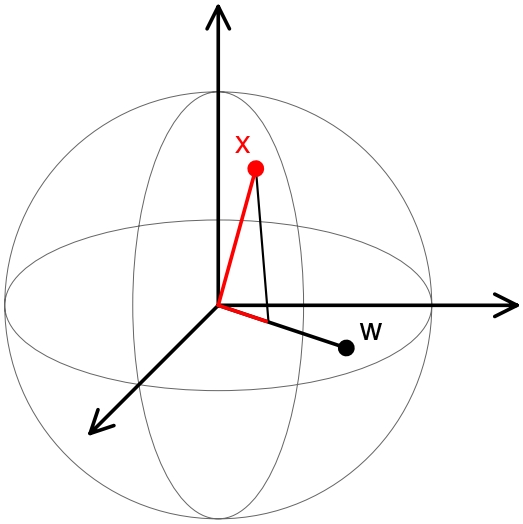

Abbildung 2.2: *Geometrische Interpretation des Skalarprodukts.*

Die Aktivierung des Neurons ist dann umso größer, je mehr der Eingangssignalvektor in die Richtung des Gewichtsvektors zeigt. Unter gewissen Bedingungen kann also ein Neuron detektieren, wie stark ein bestimmtes, durch den Gewichtsvektor festgelegtes Merkmal auf das Eingangssignal zutrifft. Mehrere solcher Neuronen können eine Signalkategorisierung vornehmen, wenn das Neuron mit der stärksten Aktivierung anzeigt, welcher Kategorie das Signal zugeordnet werden soll.

Verbindungen zwischen Neuronen

Ein neuronales Netz besteht aus einer Menge von Neuronen, die über gewichtete, gerichtete Verbindungen miteinander verknüpft sind. Die Menge der Neuronen unterteilt sich in die folgenden drei Untermengen:

1. Eingangsneuronen: Stellen dem Netz einen Sensorwert zur Verfügung und besitzen daher keine Eingänge, sondern nur einen Ausgang

2. Ausgangsneuronen: Neuronen, deren Ausgangssignal einen Aktuator ansteuert, zusätzlich aber auch als Eingangssignal für andere Neuronen dienen kann

3. Versteckte Neuronen: Neuronen, die weder zu Sensoren noch Aktuatoren Verbindungen haben, sondern nur mit anderen Neuronen verknüpft und daher nach außen nicht sichtbar sind

Ein zeitdiskretes, neuronales Netz ist somit mit der vollständigen Angabe des 6-Tupels $(n_E, n_A, n_V, \boldsymbol{x}(0), \mathbf{W}, f)$ wohldefiniert:

$$\widetilde{\boldsymbol{x}}(t+1) := f\left(\mathbf{W}\boldsymbol{x}(t)\right), \qquad t \in \mathbb{N}, \tag{2.4}$$

wobei

$$\widetilde{\boldsymbol{x}} := \begin{pmatrix} x_{n_E+1} \\ \vdots \\ x_{n_E+n_A} \\ x_{(n_E+n_A)+1} \\ \vdots \\ x_{(n_E+n_A)+n_V} \end{pmatrix}, \qquad \boldsymbol{x} := \begin{pmatrix} x_1 \\ \vdots \\ x_{n_E} \\ \widetilde{\boldsymbol{x}} \end{pmatrix}. \tag{2.5}$$

Das Netz besitzt n_E Eingangs-, n_A Eingangs- und n_V versteckte Neuronen. Der Vektor $\widetilde{\boldsymbol{x}} \in \mathbb{R}^{(n_A+n_V)}$ enthält als Einträge die Signale der Ausgangsneuronen und der versteckten Neuronen, während $\boldsymbol{x} \in \mathbb{R}^{(n_E+n_A+n_V)}$ zusätzlich um die Signale der Eingangsneuronen erweitert ist. Die Zeitindizes sind der Übersichtlichkeit halber weggelassen.

Die Gewichtsmatrix $\mathbf{W} \in \mathbb{R}^{(n_A+n_V)\times(n_E+n_A+n_V)}$ besteht aus den Verbindungsgewichten (w_{ij}), wobei w_{ij} der Verbindung vom Ausgang des Neurons j zum Eingang des Neurons i zugeordnet ist. In der Regel sind nicht alle Verbindungen vorhanden, so dass $\mathbf{W}$ an den entsprechenden Stellen Nullen eingetragen hat. Der Vektor $\mathbf{W}\boldsymbol{x}(t)$ enthält die Aktivitäten der Ausgangsneuronen und der versteckten Neuronen zum Zeitpunkt t. Die Transferfunktion $f : \mathbb{R} \to \mathbb{R}$ wird elementweise auf ihn angewandt.

Bias-Neuron

Über die Verbindungsgewichte lässt sich der Signalfluss verändern. Ist das Gewicht größer als Eins, wird das Signal verstärkt, zwischen Null und Eins wird es abgeschwächt. Negative Werte bewirken eine Phasenumkehr und eine Null bedeutet, dass keine Verbindung vorhanden ist. Die gewichtete Summe der Eingangssignale – mit anderen Worten die interne Neuronenaktivität – pendelt jedoch stets um einen gewissen Mittelwert.

Je nach Art der Transferfunktion f kann es notwendig sein, diese mittlere Aktivität mit einem Offset zu versehen, damit ein gewünschter Arbeitspunkt von f voreingestellt ist. Manchmal sind auch die Sensorsignale selbst mit einem unerwünschten Offset behaftet, der ausgeglichen werden muss.

Zu diesem Zweck kann man vorsehen, dass das Eingangsneuron N_1 konstant den Wert Eins liefert. Bei Bedarf kann die mittlere Aktivierung eines Neurons angehoben oder abgesenkt werden, indem das Verbindungsgewicht von N_1 entsprechend gesetzt wird. Das Neuron N_1 nennt man dann Bias-Neuron.

2.2 Nicht-lineare Transferfunktion

Die am häufigsten verwendeten Transferfunktionen sind neben der Identität die Sprungfunktion, die Standardsigmoide und der Tangens Hyperbolicus, wie in Abbildung 2.3 zu sehen.

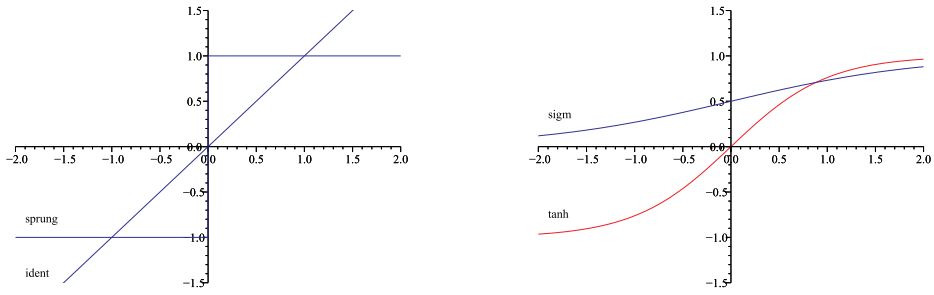

Abbildung 2.3: *Identität, Sprungfunktion, Standardsigmoide und Tangens Hyperbolicus sind die am häufigsten verwendeten Transferfunktionen.*

Verwendet man die Identität $ident(x) := x$ als Transferfunktion, dann fällt die Theorie der neuronalen Netze vollständig auf die lineare Algebra zurück. Dies bedeutet zwar eine starke Einschränkung der Netzdynamik, doch

lassen sich auch damit noch interessante Probleme lösen. Insbesondere bleiben lineare Signalfilter (FIR- und IIR-Filter) implementierbar.

Die Sprungfunktion

$$sprung(x) := \begin{cases} +1 & \text{für } x > 0 \\ -1 & \text{für } x \leq 0 \end{cases} \tag{2.6}$$

wurde bereits von McCulloch und Pitts [MP43] verwendet (allerdings mit dem Wertebereich $\{0, 1\}$ anstelle von $\{-1, 1\}$). Stellt man einen bestimmten Bias-Wert β ein, dann funktioniert sie wie ein binärer Schalter, der das Neuron aktiv werden lässt, wenn die gewichtete Summe der Eingangssignale (ohne Bias) den Wert β übersteigt. Für kognitive Systeme ist eine derartige Komparatorfunktion von Interesse, weil sich damit Wahrnehmungsschwellen realisieren lassen.

Vom Funktionsverlauf her gesehen liegen zwischen *ident* und *sprung* die sigmoiden Transferfunktionen, zu denen neben der Standardsigmoide

$$sigm(x) := \frac{1}{1 + \mathrm{e}^{-x}} \tag{2.7}$$

auch der Tangens Hyperbolicus

$$\tanh(x) := \frac{2}{1 + \mathrm{e}^{-2x}} - 1 \tag{2.8}$$

gehört. Über die Gleichung

$$sigm(x) = \frac{\tanh(x/2) + 1}{2} \tag{2.9}$$

lassen sich nicht nur die Sigmoiden selbst leicht ineinander überführen, sondern auch neuronale Netze, welche die eine oder andere Sigmoide als Transferfunktion verwenden. Eine ausführliche Darstellung hierzu findet man in [Pas02]. Der Tangens Hyperbolicus bietet aufgrund seiner Punktsymmetrie zum Ursprung

$$\tanh(-x) = -\tanh(x) \tag{2.10}$$

zwei Vorteile gegenüber der Standardsigmoiden: Erstens stellen sich Hystereseeffekte auch ohne Bias ein, da bereits Ursprungsgeraden den Tangens Hyperbolicus in drei Punkten schneiden können, also die Gleichung $x = \tanh(wx)$ drei Lösungen besitzt (für $w > 1$, siehe hierzu Kapitel 3.1).

Zweitens ist der Tangens Hyperbolicus rechentechnisch oft einfacher zu handhaben und wird daher im weiteren stellvertretend für die Klasse aller sigmoiden Transferfunktionen betrachtet.

Die beiden Transferfunktionen *ident* und *sprung* lassen sich auf einfache Weise mit Hilfe des Tangens Hyperbolicus approximieren. Betrachtet man die Abbildung

$$f_{a,w}(x) := a \tanh(wx) \tag{2.11}$$

für $w \ll 1$, $a = w^{-1}$, dann gilt $f_{a,w}(x) \approx x$. Man erhält also die Identität, wenn man das Signal vor dem Neuron stark abschwächt und hinter dem Neuron wieder reziprok zur Abschwächung verstärkt.

Andererseits nähert sich der Tangens Hyperbolicus für wachsendes w der Sprungfunktion an. Es gilt

$$\lim_{w \to \infty} f_{1,w}(x) = sprung(x) \qquad \text{für } x \in \mathbb{R} \setminus \{0\}. \tag{2.12}$$

An der Stelle Null ist stets $f_{1,w}(0) = 0 \neq -1 = sprung(0)$, doch dies stört nicht bei praktischen Anwendungen.

Eigenschaften

Was macht die Verwendung des Tangens Hyperbolicus als nicht-lineare Transferfunktion so interessant für neuronale Netze?

Zunächst hat er gewisse universelle Eigenschaften, da beliebige andere Transfercharakteristiken approximiert werden können. Für die Identität und die Sprungfunktion wurde dies eben gezeigt, für den allgemeinen Fall findet man Beweise und Anwendungen beispielsweise in [HSW89], [Jon90], [PG90], [Cyb89], [MM92], [Bar93], und [Ham].

Desweiteren ist er auf ganz $\mathbb{R}$ definiert, stetig differenzierbar, streng monoton steigend, umkehrbar und im Wertebereich auf das offene Intervall $(-1, +1)$ begrenzt. Da die Umkehrfunktion existiert und eindeutig ist, kann man bei der Analyse eines neuronalen Netzes die Zeit rückwärts laufen lassen, wodurch stabile und instabile Fixpunkte sich vertauschen (siehe Kapitel 3.1).

Für $|x| > 10$ kann man den Funktionswert auf ± 1 setzen, was die numerische Implementierung vereinfacht. Es macht aus diesem Grund auch keinen Sinn, mit beliebig großen Verbindungsgewichten zu arbeiten. Für die meisten Anwendungen reicht $|w_{i,j}| \leq 4$ aus und für alle anderen Fälle $|w_{i,j}| \leq 8$. Hat man bei einem Mikroprozessor nur eine geringe Bit-Breite zur Verfügung, so genügen 3–4 Vorkommastellen und die restlichen Bits gehen zugunsten der Genauigkeit (siehe hierzu Tabelle 11.1 auf Seite 118 zur Rechengenauigkeit beim TED).

Über die Größe der Verbindungsgewichte lässt sich steuern, ob ein neuronales Netz eher als lineares System arbeitet, oder im Sättigungsbereich.

Die Verzerrungen, welche ein Signal durch die Sättigung erfährt, lassen sich genau quantifizieren. Hierzu betrachtet man mit

$$f_{1,w}(\sin(2\pi x)), \qquad \text{für } w > 0 \tag{2.13}$$

das Frequenzspektrum einer durch den Tangens Hyperbolicus verzerrten Sinusschwingung bei wachsendem w. Einen ersten Eindruck der nicht-linearen Verzerrungen bekommt man durch Abbildung 2.4.

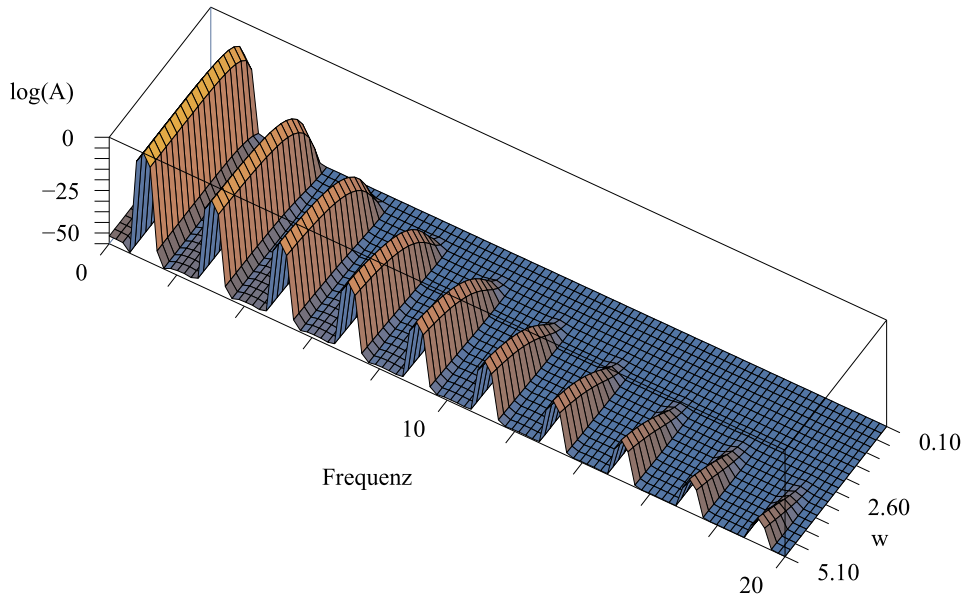

Abbildung 2.4: *Frequenzspektrum einer durch den Tangens Hyperbolicus verzerrten Sinusschwingung in Abhängigkeit des Gewichts w. Die Amplituden sind logarithmiert und die Frequenzen relativ zur Grundschwingung angegeben.*

Man kann gut erkennen, wie sich das Frequenzspektrum relativ schnell dem einer Rechteckschwingung mit 50% Tastzyklus annähert. Aufgrund der bereits erwähnten Punktsymmetrie zum Ursprung (Gleichung 2.10) sind nur die ungeradzahligen Vielfachen der Grundfrequenz vorhanden. Das Spektrum einer exakten Rechteckschwingung lässt sich einfach an ihrer Fourier-Zerlegung ablesen:

$$rechteck(x) = \sum_{i=0}^{\infty} \frac{\sin\left((2i+1)x\right)}{2i+1}. \tag{2.14}$$

Prinzipiell lassen sich mit neuronalen Netzen beliebige periodische Funktionen annähern. Wie man eine Sinusschwingung beliebiger Frequenz erzeugt, wird ausführlich in Kapitel 4.1 gezeigt. Durch gezielte nicht-lineare Verzerrung des Sinus erhält man dann die gewünschte Schwingungsform. Hierfür bedient man sich der Chebyshev-Polynome T_n, welche eine Sinusschwingung der Kreisfrequenz ω_0 auf die n-ten Harmonischen abbilden. Die rekursive Definition der Polynome lautet wie folgt:

$$\begin{aligned} T_0(x) &= 1, \\ T_1(x) &= x, \\ T_n(x) &= 2xT_{n-1}(x) - T_{n-2}(x). \end{aligned} \tag{2.15}$$

Alle Polynome bilden das Intervall $[-1, +1]$ auf sich selbst ab, wobei das Polynom n-ten Grades T_n genau n Nullstellen besitzt. Abbildung 2.5 zeigt die ersten sechs Polynome T_0 bis T_5.

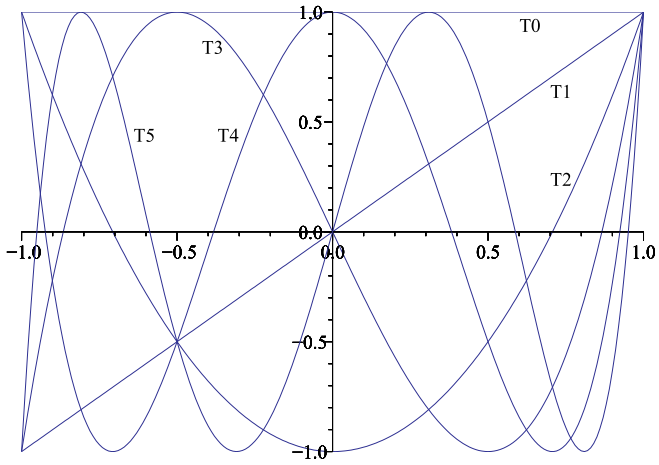

Abbildung 2.5: *Die ersten sechs Chebychev-Polynome.*

Die Verzerrungsfunktion für ein gewünschtes Spektrum erhält man durch gewichtete Summierung der entsprechenden Chebychev-Polynome. Bei gegebenen Koeffizienten h_k von $k = 1, \ldots, n$ Harmonischen, lassen sich die n Koeffizienten a_k des zugehörigen Transferpolynoms f in geschlossener Form

angeben:

$$f(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n, \tag{2.16}$$

$$a_k = 2^{k-1} \sum_{j=0}^{\lfloor (n-k)/2 \rfloor} \frac{(-1)^j (k+2j)(k+j-1)!}{k!\; j!} \cdot d_{k+2j}. \tag{2.17}$$

Dieses Verfahren ist in der Computermusik auch unter dem Begriff *Waveshaping* bekannt und lässt sich zur Erzeugung akustischer Klänge verwenden, wie sie beispielsweise der Do:Little (siehe Kapitel 10.2) als hörbares Signal ausgeben kann. Details zum Verfahren und seiner praktischen Anwendung findet man in [Bea79a] und [Bea79b].

Implementierung

Will man den Tangens Hyperbolicus als Transferfunktion implementieren, dann kann man mehrere Wege einschlagen. Wenn die notwendige Rechenleistung zur Verfügung steht, lässt sich der Tangens Hyperbolicus über den Cody-Waite-Algorithmus und ein paar Fallunterscheidungen bis auf 60 Bits genau approximieren, wie in [Bee93] gezeigt wird.

Bei einem autonomen Roboter soll die Implementierung hingegen oft auf einem 8- oder 16-Bit-Mikroprozessor erfolgen – dann kostet eine hohe Rechengenauigkeit zu viel kostbare Prozessorzeit. Glücklicherweise sind die meisten neuronalen Netze sehr robust gegenüber einer approximierten Transferfunktion, so dass in der Regel eine stückweise lineare Approximation zulässig ist.

Diese Lösung wurde auch beim TED (Kapitel 11.2) gewählt. Da der verwendete 8-Bit-Mikrocontroller mit 1.6MHz betrieben wird, keinen Arbeitsspeicher und lediglich 1kByte Programmspeicher besitzt, standen für die Transferfunktion nur fünf Geradensegmente zur Verfügung. Die Abknickpunkte der Segmente wurden mit Hilfe einer künstlichen Evolution optimiert. Das Resultat zeigt Abbildung 2.6 auf der nächsten Seite. Die exakte Definition von f ist durch folgende Gleichungen gegeben:

$$f(x) := \begin{cases} f_+(x) & \text{für } x \geq 0 \\ -f_+(-x) & \text{für } x < 0 \end{cases}, \tag{2.18}$$

$$f_+(x) := \begin{cases} m_0 x & \text{für } x \leq x_0 \\ m_1 x + b_1 & \text{für } x_0 < x < x_1 \\ 1 & \text{für } x \geq x_1 \end{cases}, \tag{2.19}$$

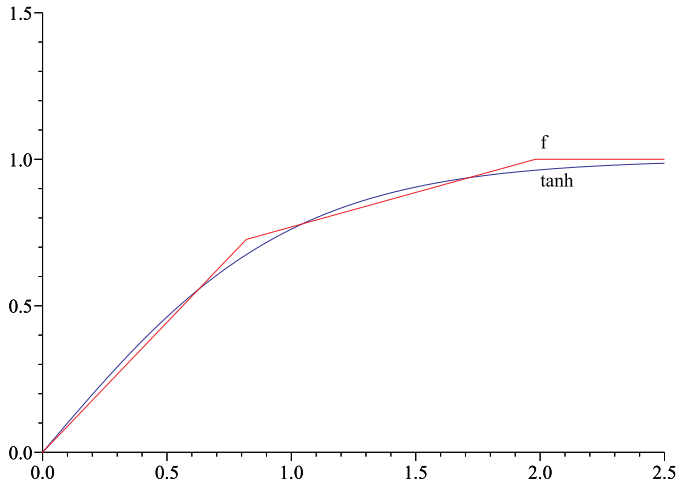

Abbildung 2.6: *Approximation der Transferfunktion Tangens Hyperbolicus durch mehrere Geradensegmente.*

wobei

$$\begin{aligned} m_0 &= 0.8872, \quad x_0 = 0.81881, \\ m_1 &= 0.2354, \quad x_1 = 1.98088, \\ b_1 &= 0.5337. \end{aligned}$$

Aufgrund der begrenzten Rechengenauigkeit des TED (siehe hierzu Tabelle 11.1 auf Seite 118), genügt es, die Parameter auf vier beziehungsweise fünf Stellen genau anzugeben.

Beim Experiment zur Phonotaxis (Kapitel 9.4) wäre die Verwendung von fünf Geradensegmenten zu ungenau gewesen, da der dort verwendete digitale Signalprozessor (DSP) eine 24-Bit-Festpunktdarstellung verwendet. Es wurde daher auf eine Funktionstabelle mit 512 Einträgen zurückgegriffen, aus welcher die Werte mit linearer Interpolation entnommen wurden. Letztere kostet den DSP nur wenige Taktzyklen.

Stückweise lineare Interpolation einer sigmoiden Transferfunktion wird auch häufig eingesetzt, wenn das neuronale Netz nicht auf einem allgemeinen Mikroprozessor implementiert werden soll, sondern in spezieller Hardware, welche auf die Verarbeitung neuronaler Netze optimiert ist (siehe [Hik03] und [BPVL94]). Bei der Hardware handelt es sich entweder um die Programmierung eines sogenannten *Field Programmable Gate Array* (FPGA), wie beispielsweise in [Gir] beschrieben, oder um ein digitales Chipdesign.

Selbstverständlich existieren auch unterschiedliche analoge Schaltungstechniken, welche die gewünschte Sättigungscharakteristik aufweisen. So wird in [BgD+99] ein analoges Chipdesign besprochen, das in 2.4μm-CMOS-Technologie gefertigt wurde und 16 Transistoren für die Berechnung der Sigmoiden benötigt. Eine ähnliche Implementierung findet sich in [HNS92] für eine hexagonale Neuronenanordnung. Schließlich gibt es auch Designstudien für hoch integrierte Chips. Das in [MH03] beschriebene System vereint 128 Neuronen mit über 2048 Synapsen und der notwendigen Ansteuerung auf einem in 0.35μm-Technologie realisierten Chip. Die eingesetzte Nicht-Linearität ist dort allerdings quadratisch.

Kapitel 3

Dynamik rekurrenter Netze

Die Verbindungen eines neuronalen Netzes bilden einen gerichteten Graphen. Enthält dieser Zyklen, so spricht man von einem *rekurrenten* Netz, ansonsten von einem *Feed-Forward*-Netz. Während sich die bisherigen Betrachtungen von nicht-linearen Eigenschaften stets auf Feed-Forward-Netze bezogen haben, stehen im weiteren die Effekte zur Diskussion, welche sich nur in rekurrenten Strukturen finden lassen.

3.1 Betrachtungen am Einzelneuron

Die kürzeste herstellbare, rekurrente Verbindung in einem neuronalen Netz ist die Selbstkopplung eines Neurons. Legt man die Definition aus Gleichung 2.4 zugrunde und betrachtet dabei ein bestimmtes Neuron N_i, das nicht Eingangsneuron ist ($i > n_E$), dann wird die Selbstkopplung durch das Gewicht $w_{i,i+n_E}$ beschrieben. Lokal, von N_i aus gesehen hat man

$$x(t+1) := \tanh\left(wx(t) + b\right), \tag{3.1}$$

wobei

$$w := w_{i,i+n_E}, \qquad b := \sum_{j=1}^{n_E+n_A+n_V} \left(1 - \delta_{j,i+n_E}\right) w_{ij} x_j(t). \tag{3.2}$$

Alle Signale von anderen Neuronen sowie ein eventueller Bias-Wert sind als Aktivierung in b zusammengefasst, während die Selbstkopplung im folgenden kurz mit w bezeichnet wird. Analysiert man nun das Verhalten des Neurons in Abhängigkeit von (w, b) für $t \to \infty$, so ergibt sich die in Abbildung 3.1 zu sehende Landschaft.

Für $w = 0$ zeigt sich der Tangens Hyperbolicus in Abhängigkeit von b und unabhängig vom Startwert $x(0)$. Mit negativ werdender Selbstkopplung

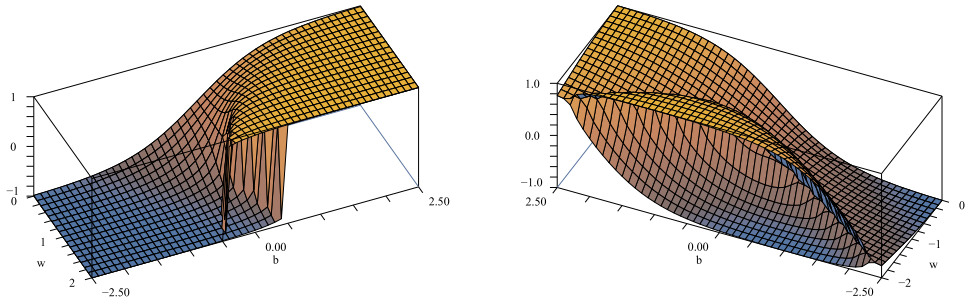

Abbildung 3.1: *(links) Stark positive Selbstkopplungen, $w > 1$, bewirken ein Hystereseverhalten des Neurons. (rechts) Bei stark negativen Selbstkopplungen, $w < 1$, oszilliert die neuronale Aktivität zwischen zwei Werten. Man kann sich die beiden Grafiken an der Rückseite ($w = 0$) zusammengeheftet vorstellen.*

$w < 0$ reduziert sich zunächst die Steilheit der Transferfunktion. Ab $w < -1$ öffnet sich um die Stelle $b = 0$ ein Bereich, wo das Neuron zwischen zwei Werten zu oszillieren beginnt. Je negativer w ist, desto größer wird dieser Bereich und desto näher rücken die Funktionswerte an ± 1. Für negativ werdende w ergibt sich also zunächst ein stabiler Fixpunkt, der nach der Bifurkation instabil und von einem Periode-2-Orbit umschlossen wird.

Hysterese

Ein völlig anderes Bild zeigt sich für positive w. Hier nimmt die Steilheit der Transferfunktion zunächst zu und ab $w > 1$ zeigt sich ein Hystereseverhalten. Der stabile Fixpunkt wird nach der Bifurkation instabil und von zwei stabilen Fixpunkten eingeschlossen. Welcher der beiden Fixpunkte angenommen wird, hängt davon ab, auf welcher Seite des instabilen Fixpunkts der Startwert lag.

Abbildung 3.2 zeigt dies im Detail für die Selbstkopplung $w = 2.5$. Die blaue Kurve entspricht einem Schnitt senkrecht zur w-Achse der Abbildung 3.1, links. Die rote Kurve zeigt die instabilen Fixpunkte. Man erhält sie, indem man bei der Simulation die Zeit rückwärts laufen lässt (siehe Kapitel 2.2 auf Seite 14), also die Umkehrabbildung von Gleichung 3.1 iteriert.

Zu gegebener Selbstkopplung $w > 1$ lassen sich die Bifurkationspunkte b_+, b_- wie folgt berechnen. Man geht zunächst zur Umkehrabbildung von Gleichung 3.1 über, weil es sich bequemer rechnen lässt. Dies vertauscht zwar stabile mit instabilen Fixpunkten, hat aber keinen Einfluss auf den Bifurkationspunkt. Man sucht demnach den Fixpunkt $y^* \in \mathbb{R}$ von

$$y^* = \frac{\tanh^{-1}(y^*) - b}{w}, \tag{3.3}$$

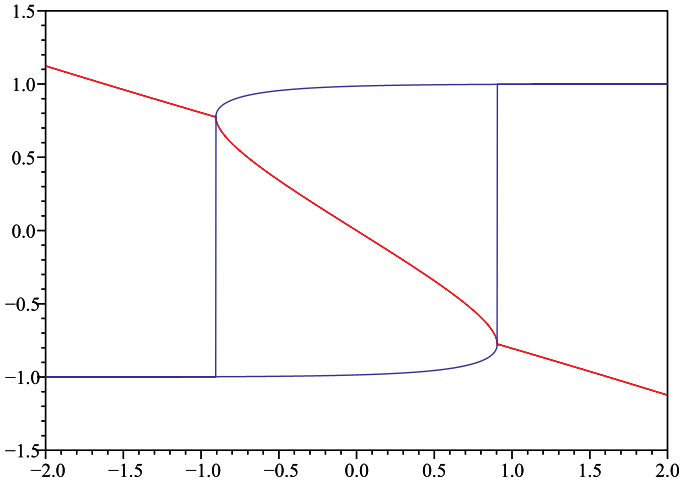

Abbildung 3.2: *Hystereseverhalten (blau) des Neurons bei einer Selbstkopplung von $w = 2.5$. Lässt man die Zeit in der Simulation rückwärts laufen, so erhält man auch die instabilen Fixpunkte (rot).*

was sich weiter umformen lässt zu

$$y^* = \frac{1}{2w} \ln \frac{1+y^*}{1-y^*} - \frac{b}{w}, \tag{3.4}$$

$$= \frac{1}{2w} [\, \ln(1+y^*) - \ln(1-y^*) \,] - \frac{b}{w}. \tag{3.5}$$

Das System ist stabil, wenn die Ableitung des rechten Terms betragsmäßig kleiner Eins ist. Am Bifurkationspunkt bricht die Stabilität gerade zusammen, daher setzt man

$$\left| \left(\frac{1}{2w} [\, \ln(1+y^*) - \ln(1-y^*) \,] - \frac{b}{w} \right)' \right| = 1, \tag{3.6}$$

$$\frac{1}{2w} \left(\frac{1}{1+y^*} - \frac{-1}{1-y^*} \right) = \pm 1, \tag{3.7}$$

$$\frac{1}{2w} \cdot \frac{1-y^*+(1+y^*)}{1-(y^*)^2} = \pm 1, \tag{3.8}$$

$$1-(y^*)^2 = \pm \frac{1}{w}, \tag{3.9}$$

$$y^* = \sqrt{1-\frac{1}{w}}. \tag{3.10}$$

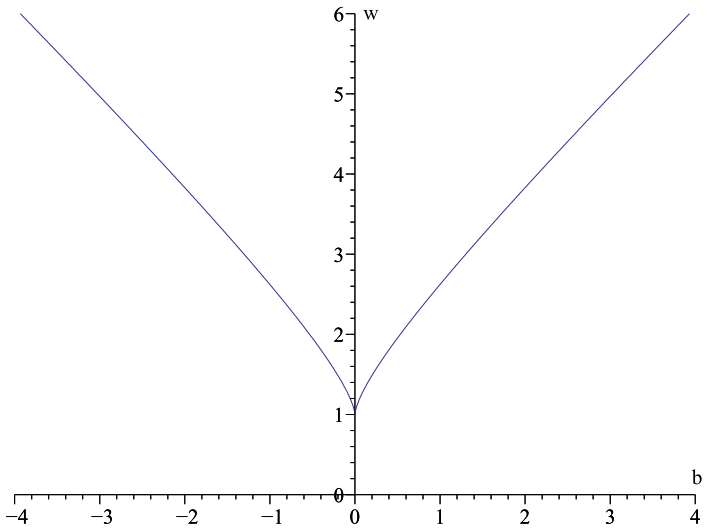

Abbildung 3.3: *Die Bifurkationsmenge des positiv rückgekoppelten Neurons wird als Cuspe bezeichnet.*

Der erste Fall kann ausgeschlossen werden, da y^* im offenen Intervall $(-1, +1)$ liegen muss. Man setzt nun y^* in Gleichung 3.4 ein, formt um und erhält

$$b_- = \ln\left(\sqrt{w} + \sqrt{w-1}\right) - \sqrt{w(w-1)}, \qquad b_+ = -b_-. \tag{3.11}$$

Für $w = 2.5$ berechnet man $b_+ \approx 0.90$, was sich auch in Abbildung 3.2 ablesen lässt. Trägt man w über b ab, so zeigt sich die sogenannte *Cuspe* als Bifurkationsmenge (siehe Abbildung 3.3).

Die theoretische Hysteresebreite $\Delta b := |b_+ - b_-|$ ist in der Praxis nur bedingt von Bedeutung, da sich b in der Regel niemals für $t \to \infty$ konstant verhält, sondern von Sensorsignalen abhängt, die sich ständig verändern. Das Gesamtsystem hält sich daher die meiste Zeit in Transienten auf und erreicht den theoretisch berechneten Fixpunkt nicht unbedingt. Dass in solchen Fällen Hystereseeffekte auch schon für $w < 1$ beobachtet werden können, zeigt Abbildung 3.4. Hier wurde das neuronale Netz nach jeder Veränderung von b nur einmal neu berechnet, wie dies auch bei autonomen Robotern gemacht wird. Bei Werten ab $w = 0.5$ beginnt der Hystereseeffekt deutlich sichtbar zu werden – die Abbildung zeigt ihn für $w = 0.9$.

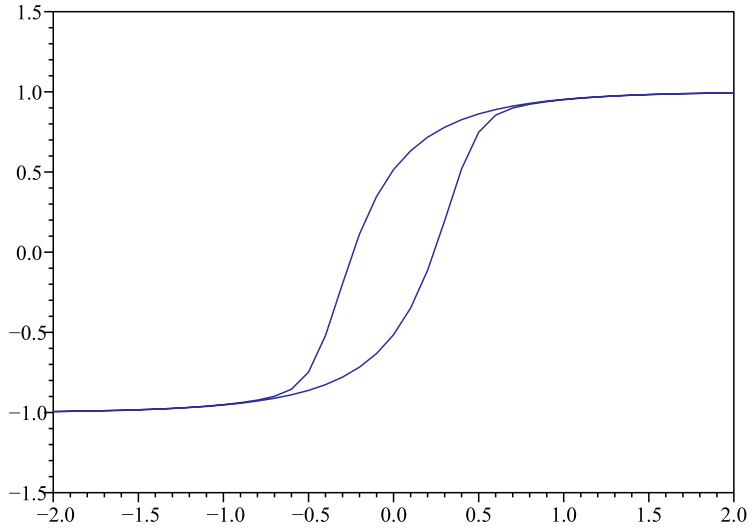

Abbildung 3.4: *Verändert man den Parameter b schnell genug, dann lässt sich auch schon für $w = 0.9$ ein Hystereseeffekt beobachten.*

Integrator

Bei subkritischer positiver Selbstkopplung $0 < w < 1$ arbeitet das Neuron als Integrator. Abbildung 3.5 zeigt das Ausgangssignal für $w = 0.99$, wenn der Eingang b mit einem Rechteckimpuls angesteuert wird (hier also ausnahmsweise $b(t)$, abweichend von der Definition in Gleichung 3.1).

Wie man sieht, ergeben sich exponentiell verlaufende Sprungantworten am Ausgang. Betrachtet man die Antwort auf einen Diracimpuls unendlicher Amplitude zum Zeitpunkt $x(0) = \infty$, dann wird der Grund offensichtlich: Aufgrund der Beschränktheit des Tangens Hyperbolicus, hat man im nächsten Moment bereits $x(1) < 1$ und innerhalb der nächsten paar Zeitschritte fällt $x(t)$ rapide in einen Bereich, wo mit $\tanh(x) \approx x$ gerechnet werden kann. Man hat ab einem gewissen Zeitpunkt t_0 also

$$x(t) \approx x(t_0) \cdot w^{t-t_0}, \qquad t \geq t_0. \tag{3.12}$$

Für w nahe Eins fällt das Anfangsintervall $[0, t_0]$ nicht mehr ins Gewicht, so dass sich wie folgt vereinfachen lässt:

$$x(t) \approx A\mathrm{e}^{-t/\tau}, \qquad A = \tanh(x(0)), \qquad \tau = \frac{1}{\ln(1/w)}. \tag{3.13}$$

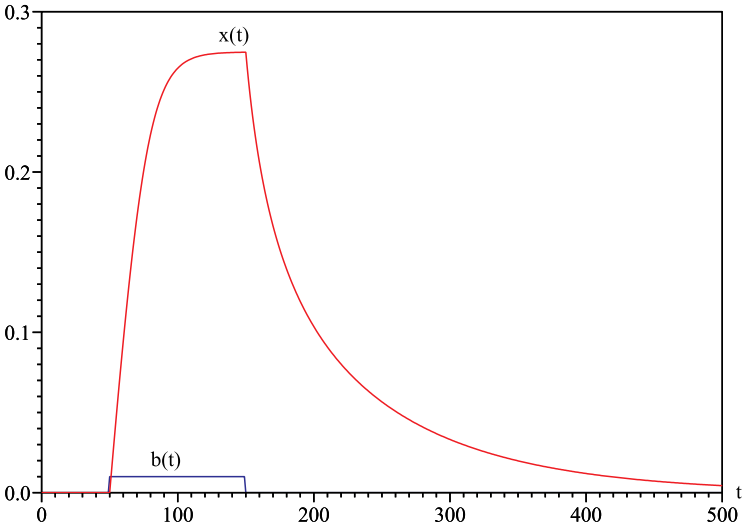

Abbildung 3.5: *Für $w = 0.99$ erhält man exponentiell verlaufende Sprungantworten am Neuronenausgang.*

Abbildung 3.6 zeigt den Verlauf des logarithmisierten Ausgangssignals $x(t)$ für $x(0) = 1$, $b = 0$ und $w = 0.92,\ 0.96,\ 0.98,\ 0.99$. Nach ungefähr 50τ Zeitschritten fällt das Ausgangssignal unter den Wert $\mathrm{e}^{-50} \approx 2 \cdot 10^{-22}$.

Filterung

Analog zum Integrator für $0 < w < 1$, verhält sich ein selbstgekoppeltes Neuron für $-1 < w < 0$ wie ein Differenziator. Es lässt sich daher hervorragend als Signalfilter einsetzen, beispielsweise wenn akustische Signale im Frequenzbereich beschnitten werden sollen. Je nach dem, ob das Vorzeichen der Selbstkopplung w positiv oder negativ ist, werden die durch das Neuron geleiteten Signale entweder mit einem Tiefpass oder einem Hochpass gefiltert.

Die zugrunde liegende Schaltung ist in Abbildung 3.7 zu sehen. Das Eingangssignal $x_e(t)$ gelangt über das Verbindungsgewicht w_e zum Neuron. Das Ausgangssignal $x_a(t)$ wird wie folgt berechnet:

$$x_a(t+1) := \tanh(w_e x_e(t) + w_a x_a(t)), \tag{3.14}$$

wobei

$$w_a := \cos\varphi, \qquad w_e := \sin\varphi, \qquad 0 < \varphi < \pi. \tag{3.15}$$

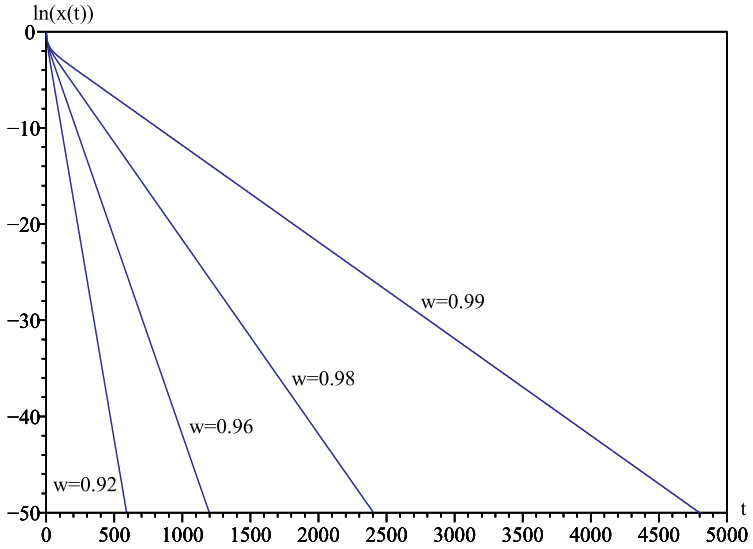

Abbildung 3.6: *Über die Selbstkopplung w lässt sich die Zeit bestimmen, die das Ausgangssignal benötigt, um einen bestimmten Grenzwert zu unterschreiten.*

Die Frequenzskala in Abbildung 3.7 versteht sich relativ zur Abtastfrequenz f_s des Eingangssignals $x_e(t)$, mit welcher auch die Aktivität des Neurons neu berechnet wird. Nach dem Nyquist-Shannonschen Abtasttheorem können maximal Frequenzen bis zu $f_s/2$ erfasst werden, weshalb die Skala bis 0.5 reicht.

Für $\varphi = \pi/2$ ist die Selbstkopplung $w_a = 0$ und das Verbindungsgewicht vom Eingang $w_e = 1$, so dass das Eingangssignal unverändert am Ausgang erscheint – sieht man von der nicht-linearen Verzerrung durch den Tangens Hyperbolicus ab. Bei wachsender Eingangsamplitude nehmen die Verzerrungen stetig zu, wie in Abbildung 2.4 auf Seite 15 gezeigt. Will man das Signal auf klassische Weise filtern, muss man also dafür sorgen, dass die Eingangsamplituden nicht zu groß werden.

Steuert man das Filterneuron mit einem Dirac-Impuls an, dann schwingt das Ausgangssignal theoretisch niemals vollständig aus. Diese Art von Filter werden in der digitalen Signaltechnik daher *Infinite Impulse Response-Filter* (kurz IIR-Filter) genannt. Praktisch gesehen wird selbstverständlich irgendwann die Rechengenauigkeit unterschritten, so dass das Signal im Systemrauschen verschwindet. Baut man mit Hilfe rekurrenter neuronaler Netze komplexere Filterstrukturen auf, muss man Betrachtungen zur Rechengenauigkeit des Gesamtsystems anstellen, da sich bei IIR-Filtern wegen der

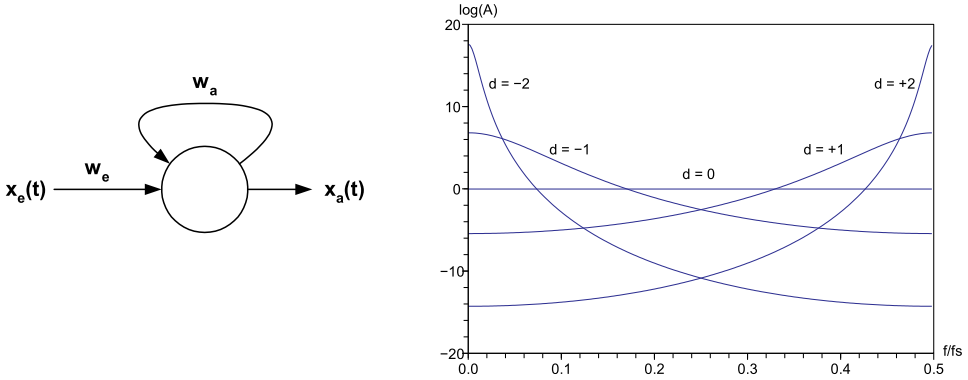

Abbildung 3.7: *(links) Modell des Filterneurons, dessen Verbindungsgewichte für* $0 < \varphi < \pi$ *durch* $w_a := \cos\varphi$, $w_e := \sin\varphi$ *definiert sind. (rechts) Übertragungscharakteristiken des Filterneurons für die Werte* $\varphi = \pi/2 + 0.6d$. *Tiefpassfilter erhält man für* $d < 0$, *Hochpassfilter für* $d > 0$.

Rückkopplungen die Rechenfehler zu ungewollten Eigenschwingungen aufschaukeln können.

3.2 Analogien zur elektronischen Schaltungstechnik

Wenn eine neuronale Steuerung für einen autonomen, mobilen Roboter eingesetzt werden soll, wird man das zugrunde liegende neuronale Netz klassischerweise auf einem Mikroprozessor implementieren. Man könnte auch auf diskrete Weise elektronische Neuronen aufbauen, von der Art wie in Kapitel 9.1 auf Seite 85 gezeigt, und diese untereinander vernetzen. Eine zusätzliche, gänzlich andere Möglichkeit eröffnet sich, wenn man Abbildung 3.8 betrachtet.

Drei neuronale Grundschaltungen, die in den vorangegangenen Seiten besprochen wurden, sind hier ihren elektronischen Schaltungsäquivalenten gegenübergestellt.

Die Komparatorfunktion wurde in Zusammenhang mit der Sprungfunktion in Gleichung 2.6 auf Seite 13 kurz erwähnt. Man erhält sie, indem man die Eingangsgewichte möglichst groß wählt, wodurch sich effektiv die Steilheit des Tangens Hyperbolicus erhöht (siehe Gleichung 2.12 auf Seite 14). Das elektronische Pendant hierzu ist ein Operationsverstärker ohne Rückkopplung. Durch die offene Schleifenverstärkung haben minimale Spannungs-

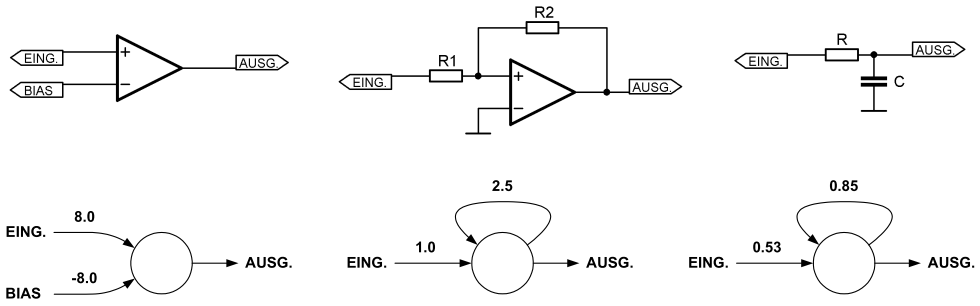

Abbildung 3.8: *Elektronische Schaltungen (oben) im Vergleich mit ihren neuronalen Entsprechungen (unten). Von links nach rechts: Komparator, Schmitt-Trigger als Hysterese-Element, Tiefpassfilter.*

differenzen zwischen Eingang und Bias positiven beziehungsweise negativen Vollausschlag am Ausgang zur Folge.

Wie beim neuronalen Hysterese-Element, kommt auch in der elektronischen Variante der Effekt durch eine positive Rückkopplung zustande. Die Breite des Hysteresefensters bestimmt sich beim abgebildeten Schmitt-Trigger relativ zum Spannungshub des Operationsverstärkers durch das Verhältnis $R1/R2$. Es gilt

$$\Delta U_{\text{hyst}} = \frac{R1}{R2} \cdot (U_{\text{amax}} - U_{\text{amin}}) \,. \tag{3.16}$$

Der als Tiefpass eingesetzte Integrator lässt sich auf einfachste Weise durch ein passives RC–Glied realisieren. Dieses Filter erster Ordnung entspricht mit seiner Flankensteilheit von 6dB/Oktave dem neuronalen Vorbild, was ein Blick auf die Filtercharakteristiken in Abbildung 3.7 bestätigt. Die Zeitkonstante des RC–Glieds beträgt $\tau = RC$ und entspricht exakt dem Parameter τ, der beim neuronalen Integrator in Gleichung 3.13 angegeben wurde.

Wie man sieht, lassen sich Einzelneuronen bei bestimmten Gewichtskonstellationen funktional direkt, also ohne den Umweg über simulierte Neuronen in elektronische Schaltungen übersetzen. Diese Methode wird bisher noch nicht angewandt – einerseits weil man die Freiheit haben möchte jederzeit Verbindungsstrukturen zu ändern, oder Lernregeln zu verwenden, welche die Gewichte zur Laufzeit modifizieren; andererseits lassen sich neuronale Netze nicht immer einfach in funktionale, in sich geschlossene Untermodule zerlegen. Trotzdem bleibt die Möglichkeit interessant, diesen Weg zu beschreiten und mit einer sehr kompakten, energieeffizienten Lösung zu enden.

3.3 Bifurkation beim 2-Neuronen-Netz

Im folgenden wird die Menge aller 2-Neuronen-Netze ohne Eingangsneuronen mit dem Tangens Hyperbolicus als Transferfunktion betrachtet. Schreibt man die Definition aus Gleichung 2.4 hierfür explizit auf, so erhält man

$$\boldsymbol{x}(t+1) := \tanh\left(\mathbf{W}\boldsymbol{x}(t)\right), \qquad \boldsymbol{x(t)} := \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix}, \qquad t \in \mathbb{N}. \tag{3.17}$$

Die Matrix der Verbindungsgewichte hat hierbei die Gestalt

$$\mathbf{W} := \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{pmatrix}, \qquad w_{ij} \in \mathbb{R}, \tag{3.18}$$

worin w_{11}, w_{22} die Selbstkopplungen darstellen und w_{12}, w_{21} die Ringkopplungen des in Abbildung 3.9 gezeigten 2-Neuronen-Netzes.

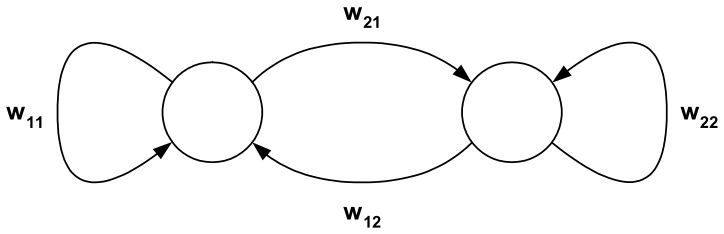

Abbildung 3.9: *2-Neuronen-Netz mit je zwei Selbstkopplungen und Ringkopplungen.*

Um Kriterien für die Schwingneigung des Netzes herleiten zu können, untersucht man die Stabilitätseigenschaften von Fixpunkten des Systems, das heißt man sucht $\boldsymbol{x}^*$, für die $\boldsymbol{x}^* = f_{\mathbf{W}}(\boldsymbol{x}^*) := \tanh(\mathbf{W}\boldsymbol{x}^*)$ gilt und analysiert dann die Eigenwerte $\lambda_{1,2}(\boldsymbol{x}^*)$ der Jacobi-Matrix $Df_{\mathbf{W}}(\boldsymbol{x}^*)$. Sie ist gegeben durch

$$Df_{\mathbf{W}}(\boldsymbol{x}^*) = \begin{pmatrix} w_{11}\tanh' x_1^* & w_{12}\tanh' x_2^* \\ w_{21}\tanh' x_1^* & w_{22}\tanh' x_2^* \end{pmatrix}. \tag{3.19}$$

Die Eigenwerte erhält man, indem man die Nullstellen des charakteristi-

schen Polynoms bestimmt:

$$|Df_{\mathbf{W}}(\boldsymbol{x}^*) - \lambda(\boldsymbol{x}^*)\mathbf{I}| \stackrel{!}{=} 0, \tag{3.20}$$

$$\left| \begin{pmatrix} w_{11}\tanh' x_1^* - \lambda(\boldsymbol{x}^*) & w_{12}\tanh' x_2^* \\ w_{21}\tanh' x_1^* & w_{22}\tanh' x_2^* - \lambda(\boldsymbol{x}^*) \end{pmatrix} \right| = 0, \tag{3.21}$$

$$\begin{aligned} (w_{11}\tanh' x_1^* - \lambda(\boldsymbol{x}^*))(w_{22}\tanh' x_2^* - \lambda(\boldsymbol{x}^*)) \\ -(w_{12}\tanh' x_2^*)(w_{21}\tanh' x_1^*) = 0, \end{aligned} \tag{3.22}$$

$$\lambda^2(\boldsymbol{x}^*) - \mathcal{S}(\boldsymbol{x}^*)\lambda(\boldsymbol{x}^*) + \mathcal{D}(\boldsymbol{x}^*) = 0, \tag{3.23}$$

wobei

$$\mathcal{S}(\boldsymbol{x}^*) = w_{11}\tanh' x_1^* + w_{22}\tanh' x_2^*, \tag{3.24}$$

$$\begin{aligned} \mathcal{D}(\boldsymbol{x}^*) = (w_{11}\tanh' x_1^*)(w_{22}\tanh' x_2^*) \\ - (w_{12}\tanh' x_2^*)(w_{21}\tanh' x_1^*) \end{aligned} \tag{3.25}$$

Spur und Determinante von $Df_{\mathbf{W}}(\boldsymbol{x}^*)$ darstellen. Final ergibt sich also

$$\lambda_{1,2}(\boldsymbol{x}^*) = \frac{1}{2}\left[\mathcal{S}(\boldsymbol{x}^*) \pm \sqrt{\mathcal{S}^2(\boldsymbol{x}^*) - 4\mathcal{D}(\boldsymbol{x}^*)}\right]. \tag{3.26}$$

Alles weitere hängt nun von den Eigenwerten ab. Der Fixpunkt $\boldsymbol{x}^*$ ist asymptotisch stabil für $-1 < \lambda_{1,2}(\boldsymbol{x}^*) < 1$. Wenn mindestens einer der beiden Eigenwerte absolut größer als Eins ist, handelt es sich um einen instabilen Fixpunkt. Ist ein Eigenwert genau Eins und der andere absolut kleiner als Eins, dann müssen höhere Ableitungen von $f_{\mathbf{W}}(\boldsymbol{x}^*)$ berücksichtigt werden, um genaue Aussagen machen zu können.

Am interessantesten ist der Fall, wenn die beiden Eigenwerte komplex konjugiert sind, also von der Form

$$\lambda_{1,2}(\boldsymbol{x}^*) = u \pm \mathrm{i}v, \qquad u \in \mathbb{R}, \qquad v \in \mathbb{R} \setminus \{0\}, \tag{3.27}$$

oder äquivalent in Polarkoordinaten

$$\lambda_{1,2}(\boldsymbol{x}^*) = r\mathrm{e}^{\pm \mathrm{i}\phi}, \qquad r > 0, \qquad \phi \neq 0, \pi. \tag{3.28}$$

Die beiden im Fixpunkt ansetzenden Eigenvektoren erfahren in diesem Fall eine Drehung um den Winkel ϕ und eine r-fache Streckung. Damit das System stabil schwingt, sollte also $\phi \neq 0, \pi$ und $r > 1$ sein. Für $r \leq 1$ ist das System durch den Tangens Hyperbolicus zu sehr gedämpft und die Schwingung endet für $t \to \infty$ im Fixpunkt $\boldsymbol{x}^*$. Es gilt

$$r = \sqrt{u^2 + v^2} = \sqrt{\lambda_1(\boldsymbol{x}^*)\lambda_2(\boldsymbol{x}^*)} = \sqrt{\mathcal{D}(\boldsymbol{x}^*)}. \tag{3.29}$$

Aus $r > 1$ folgt somit die erste Bedingung:

$$\mathcal{D}(\boldsymbol{x}^*) > 1. \tag{3.30}$$

Um $\phi \neq 0, \pi$ sicherzustellen, müssen die Eigenwerte echt komplex sein und somit die Diskriminante in Gleichung 3.26 negativ. Daraus ergibt sich folgende zweite Bedingung:

$$\mathcal{S}^2(\boldsymbol{x}^*) - 4\mathcal{D}(\boldsymbol{x}^*) < 0, \tag{3.31}$$

$$-4\mathcal{D}(\boldsymbol{x}^*) < -\mathcal{S}^2(\boldsymbol{x}^*), \tag{3.32}$$

$$\mathcal{D}(\boldsymbol{x}^*) > \frac{1}{4}\mathcal{S}^2(\boldsymbol{x}^*). \tag{3.33}$$

Trägt man beide Bedingungen grafisch in die durch Determinante und Spur aufgespannte Ebene ein, so ergibt sich Abbildung 3.10. Für Systeme

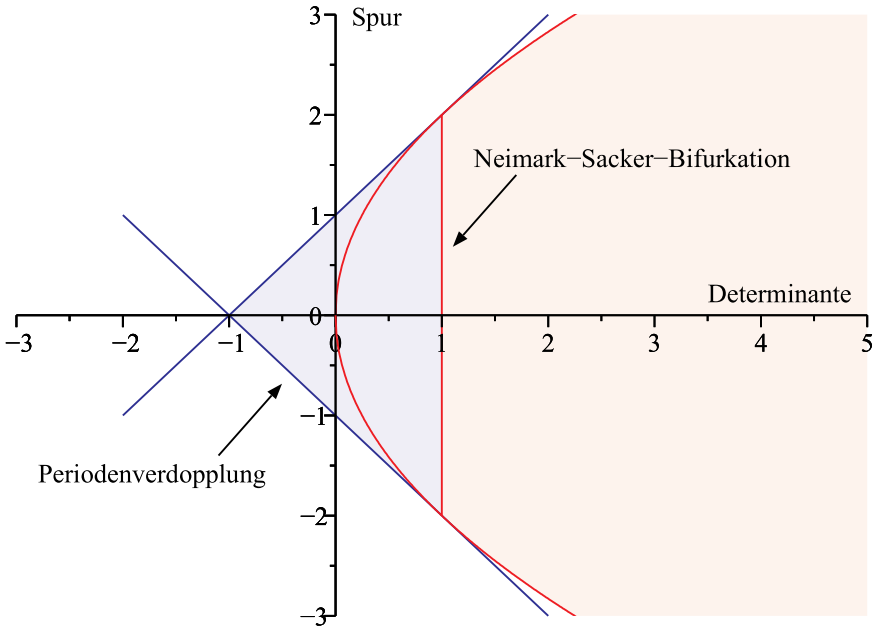

Abbildung 3.10: *Bifurkationsgrenzen in der von Determinante und Spur aufgespannten Ebene. Die Punkte der roten Fläche erfüllen die Kriterien für ein schwingendes System.*

aus dem blauen Bereich ist der Fixpunkt $\boldsymbol{x}^*$ stabil. Die drei ihn umgrenzenden Geraden stellen die Bifurkationsgrenzen dar, welche in [TS02] detailliert

diskutiert werden. Relevant ist hier die Neimark-Sacker-Bifurkation, da in ihrer Nähe die beiden Bedingungen aus den Ungleichungen 3.30 und 3.33 gleichzeitig erfüllt sind.

Wie man der Gleichung 2.4 unmittelbar ansieht, besitzt das 2-Neuronen-Netz den trivialen Fixpunkt $\boldsymbol{x}^* = \mathbf{0}$. Aufgrund von $\tanh'(0) = 1$ ergeben sich Determinante und Spur aus den Gleichungen 3.25 und 3.24 daher wie folgt:

$$\mathcal{S}(\mathbf{0}) = w_{11} + w_{22} = \text{Spur } \mathbf{W}, \tag{3.34}$$

$$\mathcal{D}(\mathbf{0}) = w_{11}w_{22} - w_{12}w_{21} = |\mathbf{W}|. \tag{3.35}$$

Die Schwingungsneigung des neuronalen Netzes lässt sich unter Berücksichtigung der Ungleichungen 3.30 und 3.33 also auf einfache Weise aus der Matrix $\mathbf{W}$ der Verbindungsgewichte direkt ableiten:

$$|\mathbf{W}| > 1, \tag{3.36}$$

$$|\text{Spur } \mathbf{W}| < 2\sqrt{|\mathbf{W}|}. \tag{3.37}$$

Mit $|\mathbf{W}| = 1 + \epsilon$, $0 < \epsilon \ll 1$ und $-2 < \text{Spur } \mathbf{W} < 2$ genügt man beiden Bedingungen und liegt unmittelbar rechts von der Neimark-Sacker-Bifurkation (siehe Abbildung 3.10). An dieser Stelle sind fast alle der im nächsten Kapitel vorgestellten Oszillatoren zu finden.

Kapitel 4

Oszillatoren als Grundlage von Motorik

Ein System, das nur sensorische Signale verarbeitet, dabei aber keinerlei Ausgangssignale produziert, hat keinen Sinn (außer vielleicht philosophischen oder künstlerischen). Ganz anders sieht es umgekehrt aus: Systeme, die ohne sensorische Informationen Ausgangssignale produzieren, können sinnvolle Aufgaben erledigen.

Unter diesen Systemen nehmen Oszillatoren eine breite Sonderstellung ein. Die Art ihrer Funktionsweise und das Spektrum der produzierten Ausgangssignale variieren beträchtlich und reichen vom 1GHz-Quarzoszillator, der mit Rechteckschwingungen den Prozessor eines PCs antreibt, bis hin zum menschlichen Sinusknoten, der mit einer Frequenz von wenigen Hertz die komplexen Steuersignale für alle Herzmuskeln generiert.

Bei Lebewesen sind Oszillatoren insbesondere Grundlage sämtlicher zyklischer Bewegungsabläufe. Ein Zyklus identifiziert dabei eine geschlossene, nicht notwendigerweise überschneidungsfreie Trajektorie im dreidimensionalen Raum. Zu ihrer Generierung bedarf es motorischer Steuersignale, die wiederum in einem n-dimensionalen Phasenraum eine Trajektorie durchschreiten.

Auch in der Robotik ist man an zyklischen motorischen Steuersignalen interessiert, beispielsweise für die Steuerung einer Roboterhand (siehe [Wil03] und Kapitel 9.2), zur Verhaltensstabilisierung (siehe [DP02]) oder auch zur Erzeugung von Laufmustern für Laufmaschinen (siehe [TTO01] und Kapitel 11). In der einfachsten, nicht sensorgetriebenen Variante werden dabei alle notwendigen Signale von einem sogenannten *Central Pattern Generator* (CPG) produziert (siehe zum Beispiel [YNA00]).

Eine Menge motorischer Signale, die phasenstarr auf der gleichen Grundfrequenz schwingen, leitet man am günstigsten von einem gemeinsamen Os-

zillator ab. Sehr geeignet ist hierfür ein Quadraturoszillator, der zwei um 90° phasenverschobene Sinusschwingungen zur Verfügung stellt. Durch Linearkombination dieser beiden Sinusschwingungen lässt sich eine Sinusschwingung derselben Frequenz, aber mit beliebiger Phasenlage und Amplitude ableiten. Über die in Kapitel 2.5 auf Seite 16 vorgestellten Chebychev-Funktionen lässt sich über nicht-lineare Verzerrungen dann die finale Kurvenform zurechtbiegen.

In den nächsten Kapiteln werden neben dem erwähnten Quadraturoszillator noch etliche weitere Oszillatoren vorgestellt, die gemeinsam haben, dass sie alle über die Bifurkation eines 2-Neuronen-Netzes erzeugt werden.

Darüber hinaus wird im Kapitel 5 ein Konzept für selbstregulierende oszillierende Neuromodule vorgestellt, welche zur Laufzeit zusammen- und auseinandergesteckt werden können und daher in motorischen Modulen eines Roboterbaukastens einsetzbar wären. Ein Beispiel dafür findet man in [Raf04].

4.1 Implementierung von Standardoszillatoren

Oszillatoren spielen in den unterschiedlichsten Bereichen eine wichtige Rolle, von naheliegenden Anwendungen wie Tonerzeugung, bis hin zur Steuerung von laufenden Robotern. Aus diesem Grund existieren bereits etliche elektronische Schaltungen sowie DSP-Algorithmen zur Erzeugung der klassischen Schwingungsformen Sinus, Rechteck und Dreieck.

Im folgenden wird gezeigt, wie fünf dieser traditionellen Oszillatoren in einem 2-Neuronen-Netz implementiert werden können. Obwohl sie aus unterschiedlichen Bereichen stammen, können sie bei neuronaler Implementierung doch in derselben Netztopologie realisiert werden – sie unterscheiden sich nur durch ihre Gewichtsmatrix $\mathbf{W}$. Da der Nullvektor ein Fixpunkt des Systems ist, wenngleich ein instabiler, wird als Startwert des Systems stets der Wert

$$\boldsymbol{x}(0) = \begin{pmatrix} 0.1 \\ 0.1 \end{pmatrix} \tag{4.1}$$

verwendet. Bei analog implementierten Varianten spielt der Startwert keine Rolle, da diese Systeme einen gewissen Rauschpegel in der Aktivität haben und sich stets stabil einschwingen, da jeder vom Nullvektor verschiedene Startwert im Attraktor landet.

SO(2)-Oszillator

Der in [Hig90] als *Coupled Form Oscillator* vorgestellte und in [PHZ03] als SO(2)-Oszillator neuronal umgesetzte Quadraturoszillator wird durch die orthogonale (für $\epsilon = 0$ orthonormale) Gewichtsmatrix

$$\mathbf{W}_2 := (1+\epsilon) \cdot \begin{pmatrix} s & r \\ -r & s \end{pmatrix}, \tag{4.2}$$

$$s := \cos(2\pi\phi), \tag{4.3}$$

$$r := \sin(2\pi\phi) \tag{4.4}$$

beschrieben, wobei $0 \leq \phi \leq 1/2$ und $0 < \epsilon \ll 1$. Er produziert zwei um $\pi/2$ verschobene Sinusschwingungen, deren Frequenz hauptsächlich von ϕ, aber auch in gewissem Maße von ϵ abhängt. Den genauen Zusammenhang zwischen der effektiven relativen Frequenz f/f_s und ϕ zeigt Abbildung 4.1 (links) für $\epsilon = 0.01$ und $\epsilon = 0.99$.

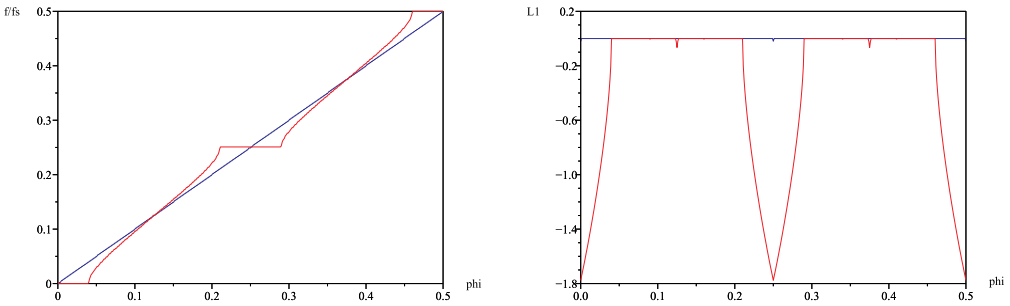

Abbildung 4.1: *Relative Frequenz f/f_s (links) und größter Lyapunov-Exponent (rechts) des SO(2)-Oszillators in Abhängigkeit von ϕ, für $\epsilon = 0.01$ (blau) und $\epsilon = 0.99$ (rot).*

Wie man sieht, ergeben sich je nach Wert von ϵ Bereiche, in denen gilt:

$$\frac{f}{f_s} = \frac{n}{4}, \qquad \text{für } |\phi - \frac{n}{4}| \leq \delta(\epsilon),\ n \in \{0,1,2\}. \tag{4.5}$$

Wie der größte Lyapunov-Exponent in Abbildung 4.1 (rechts) enthüllt, liegen in diesen Bereichen periodische Attraktoren vor, entgegen der ansonsten vorherrschenden quasi-periodischen Attraktoren. Durchfährt man mit ϕ das Intervall $[0.5, 1.0]$, dann überstreicht man erneut den kompletten Frequenzbereich, aber in umgekehrter Richtung (nämlich von den hohen zu den tiefen Frequenzen) und mit einer Phasenverschiebung von $-\pi/2$.

Bei den klassischen DSP-Implementierungen setzt man normalerweise $\epsilon = 0$, doch um die Neimark-Sacker-Bifurkationsgrenze zu überqueren, wählt man hier mit $\epsilon = 0.01$ einen minimalen positiven Wert. Der Parameter ϵ steuert im wesentlichen die Amplitude und damit zugleich auch die Verzerrung der Wellenform. Für kleine ϵ generiert das 2-Neuronen-Netz fast reine Sinusschwingungen, wohingegen sich für größere Werte annähernd Rechteckfunktionen ergeben, wie dies in Kapitel 2.2 auf Seite 15 schon diskutiert wurde. Einen Vergleich zwischen den beiden Extremen $\epsilon = 0.01$ und $\epsilon = 0.99$ bei festem $\phi = 0.047$ zeigt Abbildung 4.2. Die zugehörigen Phasendiagramme findet man in Abbildung 4.14 auf Seite 47 oben links.

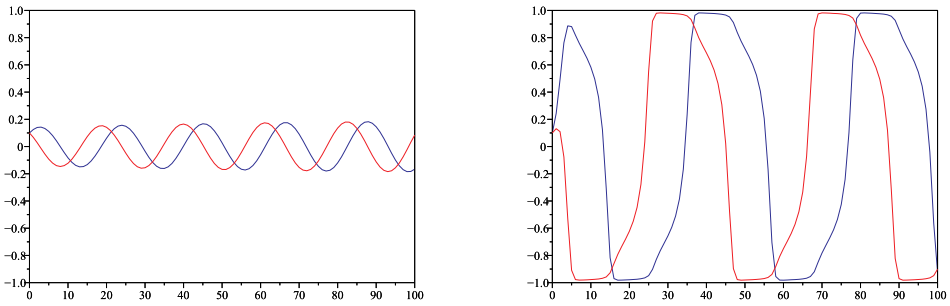

Abbildung 4.2: *Kurvenformen des SO(2)-Oszillators in Abhängigkeit von ϵ bei konstantem Wert von $\phi = 0.047$. (links) $\epsilon = 0.01$. (rechts) $\epsilon = 0.99$.*

Berechnet man Spur und Determinante der Gewichtsmatrix $\mathbf{W}_2$, so erhält man

$$\text{Spur } \mathbf{W}_2 = 2(1+\epsilon)\cos(2\pi\phi), \tag{4.6}$$

$$|\mathbf{W}_2| = (1+\epsilon)^2. \tag{4.7}$$

Wie man leicht nachrechnen kann, sind damit für $\epsilon > 0$ und $0 < \phi < 1/2$ die beiden Ungleichungen 3.36 und 3.37 erfüllt.

Neben der naheliegenden Interpretation der $\mathbf{W}_2$-Matrix als Rotationsabbildung, lässt sich die vorliegende Gewichtskonstellation auch als Hintereinanderschaltung eines Tief- und Hochpassfilters auffassen, wie in Kapitel 3.1 auf Seite 25 vorgestellt. Damit entspricht der SO(2)-Oszillator der elektronischen Schaltung eines Wien-Brücken-Oszillators, wie in Abbildung 4.3 zu sehen ist.

Die Kombination aus je zweimal R und C bildet ein passives Bandpassfilter, das aus einer Serienschaltung von Tief- und Hochpassfilter besteht (vergleiche hierzu Abbildung 3.8 auf Seite 28, rechts oben). Der Widerstand

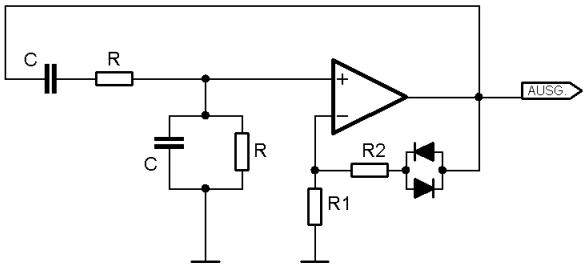

Abbildung 4.3: *Der Wien-Brücken-Oszillator besteht aus einem positiv rückgekoppelten Bandpassfilter und einem Verstärker mit Nichtlinearität.*

R gegen Masse wird hierbei nicht zwischen C und R, sondern klassischerweise parallel zu C geschaltet, was jedoch für die Resonanzfrequenz keinen Unterschied macht. Der Spannungsteiler aus $R1$ und $R2$ bestimmt den Verstärkungsfaktor und entspricht dem Term $(1+\epsilon)$ der Matrix $\mathbf{W}_2$. Die beiden antiparallel geschalteten Dioden begrenzen die Verstärkung für hohe Amplituden und erfüllen somit denselben Zweck wie die Nichtlinearität des Tangens Hyperbolicus: die Einstellung der Ausgangsamplitude. Auch für den Wien-Brücken-Oszillator gilt, dass die Verzerrungen zunehmen, wenn man den Verstärkungsfaktor erhöht.

Der SO(2)-Oszillator bildet die Grundlage der Laufroboter Lucy und TED (siehe Kapitel 11.1 und 11.2), da sich aus den phasenstarren Ausgangssignalen des Oszillators durch Linearkombination auf einfache Weise Motorsignale beliebiger Amplitude und Phasenlage generieren lassen. Abbildung 4.4 zeigt den Aktionsradius des auf diese Weise angesteuerten TEDs.

Die Resultate wurden experimentell ermittelt, indem TED im Kreismittelpunkt gestartet und für die Dauer von fünf Sekunden laufen gelassen wurde. Die Höhe der Amplitude bestimmt, wie weit TED läuft. Bei maximaler Amplitude wurde Ring C (Radius $r_C = 30\text{cm}$), bei geringeren Amplituden nur Ring B oder A erreicht ($r_B = 20\text{cm}$, $r_A = 10\text{cm}$).

Die Phasenlage zwischen vorderem und hinterem Beinpaar entscheidet zwischen Vorwärts- und Rückwärtslauf. Eine positive Phasendifferenz $\Delta\varphi = \pi/2$ zwischen vorderem und hinterem Motorsignal lässt TED vorwärts gehen, für $\Delta\varphi = -\pi/2$ läuft er rückwärts und bei $\Delta\varphi = 0$ tritt er auf der Stelle.

Da die Beinpaare im zeitlichen Mittel senkrecht zur Laufrichtung stehen, lassen sich leichte Links- und Rechtskurven erzielen, indem man die Motorsignale mit einem gegenphasigen Bias versieht. Die maximal mögliche Kurvenkrümmung erhält man, wenn die Beinpaare von oben betrachtet um $\pm 15°$ aus der Achse gedreht sind, die senkrecht zum Körper steht.

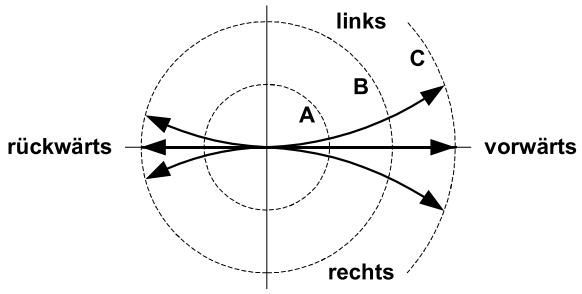

Abbildung 4.4: *Laufrichtung des TED in Abhängigkeit von Amplitude, Phasenverschiebung und Ruhepegel der Motorsignale.*

Magic Circle-Oszillator

Beim sogenannten *Magic Circle* handelt es sich um eine Variante des SO(2)-Oszillators. Er wurde in [GS85] vorgeschlagen, weil er gegenüber der klassischen Implementierung zwei Vorteile besitzt:

1. Bessere Stabilität bei begrenzter Rechengenauigkeit

2. Vereinfachte Frequenzregelung über einen linearen Parameter

Es handelt sich also um Optimierungen, die speziell für digitale Signalprozessoren und Mikroprozessoren mit Fixpunktarithmetik von Interesse sind. Eine korrespondierende elektronische Schaltung existiert nicht.

Die Gewichtsmatrix des Magic Circle lautet

$$\mathbf{W}_{MC} := (1+\epsilon) \cdot \begin{pmatrix} 1 & r \\ -r & 1-r^2 \end{pmatrix}, \tag{4.8}$$

$$r := 4\phi, \tag{4.9}$$

wobei $0 \leq \phi \leq 1/2$ und $0 < \epsilon \ll 1$ wie gehabt. Zur Steuerung der Frequenz bedarf es keiner transzendenten trigonometrischen Funktionen, da die Frequenz über einen weiten Bereich linear von ϕ abhängt, was man in Abbildung 4.5 für $\epsilon = 0.1$ bestätigt sieht. Bei größeren Werten von ϵ gibt es, wie schon beim SO(2)-Oszillator, vermehrt Bereiche von ϕ, in denen quasiperiodische Attraktoren von periodischen abgelöst werden und die Frequenz daher konstant bleibt.

Da die Steigung für $\phi \leq 0.35$ recht gering ist, hat man bei begrenzter Rechengenauigkeit eine verbesserte Auflösung im tiefen Frequenzbereich. Ist man nur an diesem Bereich interessiert, so lässt sich die Gewichtsmatrix $\mathbf{W}_{MC}$ weiter vereinfachen. Die einzige Funktion von ϵ ist, $|\mathbf{W}_{MC}| > 1$ zu

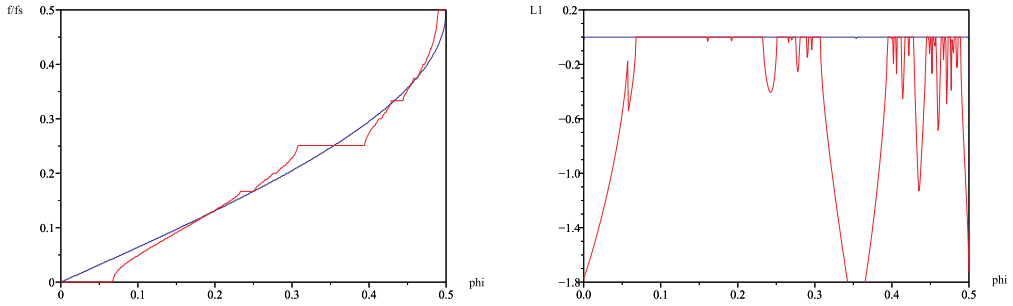

Abbildung 4.5: *Relative Frequenz f/f_s (links) und größter Lyapunov-Exponent (rechts) des Magic Circle-Oszillators in Abhängigkeit von ϕ, für $\epsilon = 0.01$ (blau) und $\epsilon = 0.99$ (rot).*

garantieren. Dies kann aber auch in der Form

$$\widetilde{\mathbf{W}}_{MC} := \begin{pmatrix} 1+\epsilon & r \\ -r & 1-r^2 \end{pmatrix} \tag{4.10}$$

erreicht werden. Bei Frequenzänderungen spart man sich auf diese Weise die Multiplikation der gesamten Matrix mit dem Faktor $(1+\epsilon)$. Abbildung 4.14 auf Seite 47 (Mitte) zeigt das Phasendiagramm des Magic Circle und seiner vereinfachten Version im Vergleich. Es gilt

$$\begin{aligned} \left|\widetilde{\mathbf{W}}_{MC}\right| &= (1+\epsilon)(1-r^2) - r(-r), && (4.11)\\ &= 1+\epsilon-\epsilon r^2, && (4.12)\\ &= 1+\epsilon(1-16\phi^2), && (4.13) \end{aligned}$$

daher nimmt die Amplitude bei steigender Frequenz ab, was für Laufmaschinen vorteilhaft ist: aufgrund der Massenträgheit können schnellere Bewegungen nur mit geringerer Auslenkung vollzogen werden.

Die Schwingung bricht vollständig ab, wenn die Neimark-Sacker-Grenze erreicht ist. Man hat also mit

$$\begin{aligned} \left|\widetilde{\mathbf{W}}_{MC}\right| &\stackrel{!}{=} 1, && (4.14)\\ 1+\epsilon(1-16\phi_{max}^2) &= 1, && (4.15)\\ 16\phi_{max}^2 &= 1, && (4.16)\\ \phi_{max} &= 1/4 && (4.17) \end{aligned}$$

maximal die untere Hälfte des Frequenzbereichs zur Verfügung. Die Berechnungsfrequenz f_s des neuronalen Netzes muss daher mindestens viermal so

hoch sein wie die maximale Laufgeschwindigkeit des Roboters. Da der TED schon mit minimaler Rechenleistung sein Netz 50-mal pro Sekunde auffrischt, stellt ϕ_{max} für ihn praktisch keine Begrenzung dar.

Biquadratischer Oszillator

Während es sich bei den letzten beiden Oszillatoren um 2-dimensionale Systeme erster Ordnung handelt, stellt der biquadratische Oszillator ein 1-dimensionales System zweiter Ordnung dar, was klar wird, wenn man sich Abbildung 4.6 ansieht: das Ausgangssignal $x(t)$ hängt direkt von $x(t-1)$ und $x(t-2)$ ab.

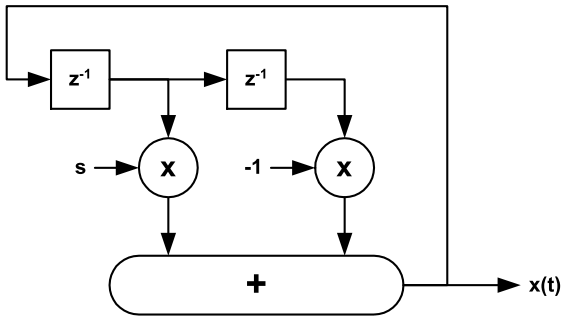

Abbildung 4.6: *Schaltungsprinzip des biquadratischen Oszillators. Die Verzögerung um eine Zeiteinheit ist mit z^{-1} bezeichnet.*

Die beiden Verzögerungsglieder bilden zusammen mit der gewichteten Summierung ein Bandpassfilter zweiter Ordnung. Aufgrund der vorhandenen Entdämpfung wird die Anordnung auch als *Direct Form Resonator* bezeichnet. Da die z-Transformation des Resonators im Zähler und Nenner quadratische Terme aufweist, ist schließlich auch der Name *Biquadratischer Oszillator* gebräuchlich (siehe auch [Hig90]).

Will man den biquadratischen Oszillator im 2-Neuronen-Netz implementieren, so platziert man die Neuronen gedanklich anstelle der Verzögerungsglieder, liest die entsprechenden Verbindungsgewichte ab und erhält

$$\mathbf{W}_{BQ} := (1+\epsilon) \cdot \begin{pmatrix} s & -1 \\ 1 & 0 \end{pmatrix}, \tag{4.18}$$

$$s := 2\cos(2\pi\phi), \tag{4.19}$$

wobei wieder $0 \leq \phi \leq 1/2$ und $0 < \epsilon \ll 1$ gilt. Es ist unmittelbar klar, dass die Ausgangssignale diesmal nicht um $\pi/2$ phasenverschoben sind, sondern lediglich um einen Zeitschritt verzögert. Für die praktische Anwendung bei

niedrigen Frequenzen existiert daher quasi nur ein einziges Signal. Im Phasendiagramm zeigt sich entsprechend anstelle eines Kreises eine sehr schmale Ellipse in Richtung der ersten Hauptdiagonalen (siehe Abbildung 4.14 auf Seite 47, unten links).

Was die Frequenzkurve des biquadratischen Oszillators betrifft, so zeigt sich in Abbildung 4.7 für $\epsilon = 0.01$ das gleiche Bild wie beim SO(2)-Oszillator. Für größere Werte von ϵ bilden sich jedoch deutlich mehr und breitere Bereiche konstanter Frequenz aus.

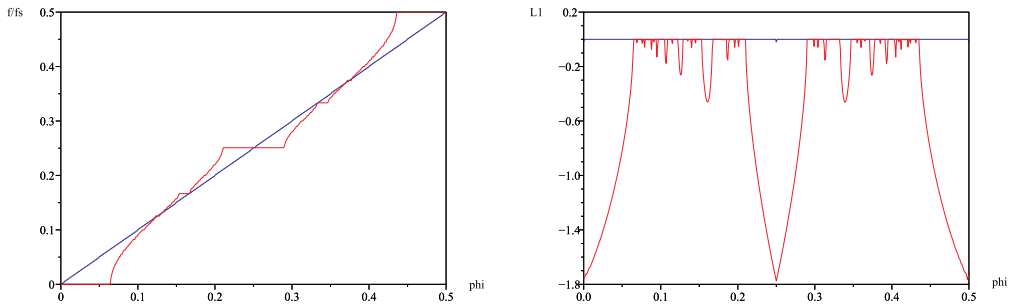

Abbildung 4.7: *Relative Frequenz f/f_s (links) und größter Lyapunov-Exponent (rechts) des biquadratischen Oszillators in Abhängigkeit von ϕ, für $\epsilon = 0.01$ (blau) und $\epsilon = 0.99$ (rot).*

Waveguide-Oszillator

Das Design des Waveguide-Oszillators wurde erstmalig in [SC92] von Smith und Cook vorgeschlagen. Es geht auf das physikalische Modell schwingender Luftsäulen zurück und ist in Abbildung 4.8 schematisch dargestellt.

Man stelle sich eine Flasche im Längsschnitt vor. Flaschenhals und Korpus sollen die gleiche Länge besitzen, aber unterschiedliche Querschnittsflächen A_1 (Korpus) und A_2 (Hals). Die Wellenimpendanz R_i in Sektion i beträgt dann $R_i = \rho c / A_i$, wobei ρ die Luftdichte und c die Schallgeschwindigkeit in der Luft ist. Für das abgeschlossene Ende gilt $R_0 = \infty$ und für die Öffnung $R_3 = 0$. Der Reflexionskoeffizient k bestimmt sich durch die Diskontinuität der Wellenimpendanzen zu

$$k = \frac{R_1 - R_2}{R_1 + R_2}. \tag{4.20}$$

An der Öffnung wird der Schall invertiert, am geschlossenen Ende hingegen gleichphasig reflektiert. An der Verbindungsstelle von Hals und Korpus wird

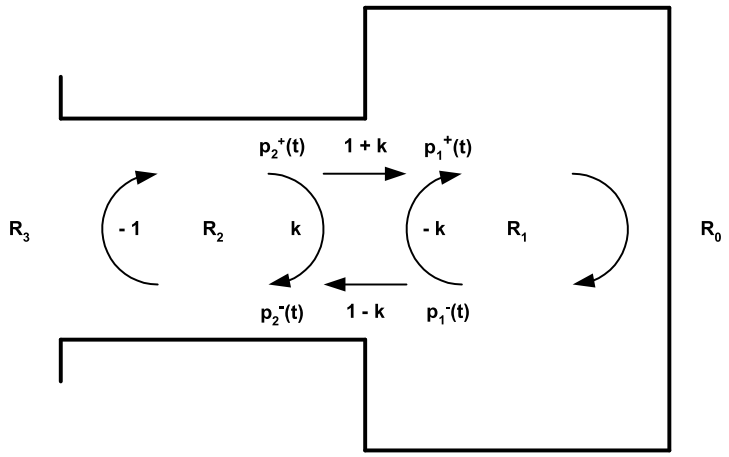

Abbildung 4.8: *Der verlustlose Waveguide-Oszillator zweiter Ordnung ist mit zwei akustischen Luftsäulen aufgebaut.*

ein Teil des Schalls übertragen und der Rest reflektiert, so dass die Energie erhalten bleibt; es liegt also eine verlustlose Streuung vor.

Übertragt man die beschriebene Flasche in eine zeitdiskrete Simulation, so ergibt sich das in Abbildung 4.9 gezeigte Modell. Die Fortpflanzungsge-

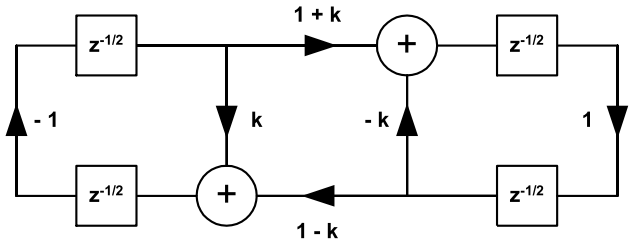

Abbildung 4.9: *Modell für die zeitdiskrete Simulation der beiden Luftsäulen aus Abbildung 4.8.*

schwindigkeit des Schalls von der Verbindungsstelle zum offenen beziehungsweise geschlossenen Ende ist jeweils mit einer halben Zeiteinheit angesetzt. In der Literatur über digitale Filter ist diese Realisierung der Verbindungsstelle als Kelly-Lochbaum Lattice-Filter bekannt. Es wird bei der Modellierung schwingender Saiten verwendet, ist aber auch häufig zur Simulation des Vokaltrakts im Rahmen künstlicher Sprachsynthese anzutreffen – eine Anwendung, die auch für die Robotik interessant ist.

Um nun das Modell neuronal umzusetzen, geht man wie folgt vor. Die beiden Verzögerungsglieder $z^{-1/2}$ am geschlossenen Ende fasst man zu einem

Glied z^{-1} zusammen. Am offenen Ende geht man ebenso vor, nachdem man die Signalinvertierung mit der Verzögerung im oberen Signalpfad vertauscht hat. Substituiert man schließlich $s := -k$, so hat man

$$\mathbf{W}_{WG} := (1+\epsilon) \cdot \begin{pmatrix} s & s+1 \\ s-1 & s \end{pmatrix}, \tag{4.21}$$

$$s := \cos(2\pi\phi), \tag{4.22}$$

mit $0 \leq \phi \leq 1/2$ und $0 < \epsilon \ll 1$ wie üblich. Den Frequenzverlauf in Abhängigkeit von ϕ zeigt Abbildung 4.10. Vergleicht man das Diagramm mit denen der vorangegangenen Oszillatoren, so zeigt sich ein deutlich eingeschränkter Regelungsbereich beim Waveguide-Oszillator. Für $\epsilon = 0.99$ setzt die Schwingung erst bei $\phi \approx 0.09$ ein und auch für $\epsilon = 0.01$ beginnt die Oszillation erst bei $\phi \approx 0.01$.

Betrachtet man die Ausgangssignale des Waveguide-Oszillators, so fällt die Amplitudendifferenz zwischen den beiden Neuronen auf: eines der Signale besitzt fast keinen Ausgangspegel, während das andere eine nahezu maximale Amplitude aufweist (siehe Abbildung 4.14).

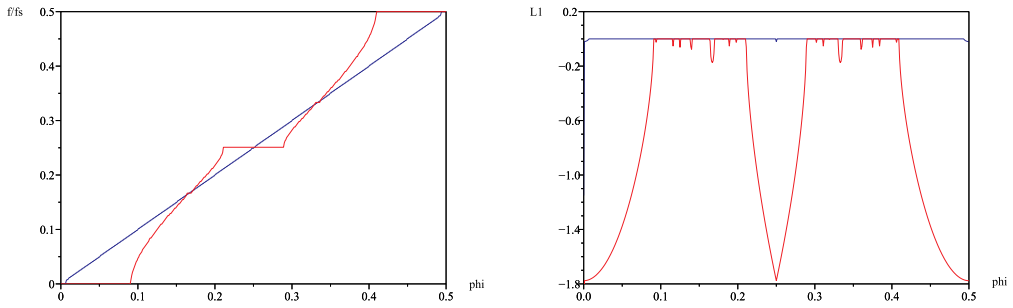

Abbildung 4.10: *Relative Frequenz f/f_s (links) und größter Lyapunov-Exponent (rechts) des Waveguide-Oszillators in Abhängigkeit von ϕ, für $\epsilon = 0.01$ (blau) und $\epsilon = 0.99$ (rot).*

4.2 Oszillator mit Integrator und Schmitt-Trigger

Schaltet man einen Integrator mit einem Schmitt-Trigger in Serie und führt das Ausgangssignal invertiert zum Eingang zurück, so erhält man einen einfachen Rechteck-Oszillator. Dieses Schaltungsprinzip ist in der Elektronik wohlbekannt. Den entsprechenden Schaltplan sieht man in Abbildung 4.11.

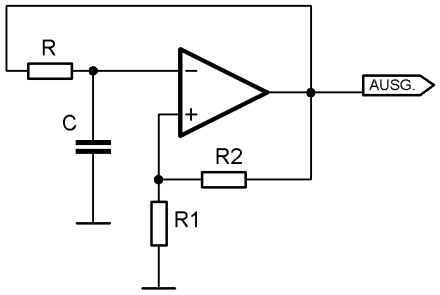

Abbildung 4.11: *Einfacher Rechteck-Oszillator mit Operationsverstärker.*

Aus schaltungstechnischen Gründen ist es einfacher, die Invertierung des Signalwegs nicht direkt am Ausgang, sondern erst nach dem Integrator am Eingang des Schmitt-Triggers vorzunehmen. Auf diese Weise kommt man mit einem einzigen Operationsverstärker aus.

Es ist möglich, Bifurkationen direkt an der elektronischen Schaltung zu studieren, wie dies beispielsweise in [KK98] und [NAA03] gemacht wurde. Andererseits kann man mit Hilfe der Analogien aus Kapitel 3.8 auf Seite 28 den Schaltplan des Rechteck-Oszillators auf einfache Weise zunächst in ein 2-Neuronen-Netz überführen. Für den Integrator benötigt man ein Neuron mit Selbstkopplung unter Eins und für den Schmitt-Trigger das Hysterese-Neuron mit einer Selbstkopplung über Eins. Zusammen mit der Invertierung ergibt sich die Gewichtsmatrix

$$\mathbf{W}_{IS} := \begin{pmatrix} 1-\alpha & -\alpha \\ 1+\beta & \beta \end{pmatrix}, \tag{4.23}$$

wobei $0 < \alpha \leq 1$ und $\beta > 1$. Der Parameter α bestimmt gemäß Gleichung 3.13 auf Seite 24 die Zeitkonstante des Integrators und korrespondiert mit der RC-Kombination im Schaltplan. Parameter β legt, wie aus Gleichung 3.4 auf Seite 22 bekannt, die Breite der Hysterese fest. In der elektronischen Schaltung wird sie analog über das Widerstandsverhältnis $R1/R2$ geregelt.

Wie man in Abbildung 4.12 sieht, hat der Parameter β nur einen geringfügigen Einfluss auf die Frequenz. Entscheidend ist, dass β nicht zu gering gewählt wird, weil die Amplitude der Schwingung sonst für niedrige Frequenzen zusammenbricht. Der Wert $\beta = 2.4$ bietet einen großen Frequenzbereich bei gleichbleibender Amplitude.

Spur und Determinante von $\mathbf{W}_{IS}$ ergeben sich zu

$$\text{Spur } \mathbf{W}_{IS} = 1 - \alpha + \beta, \tag{4.24}$$

$$|\mathbf{W}_{IS}| = \alpha + \beta. \tag{4.25}$$

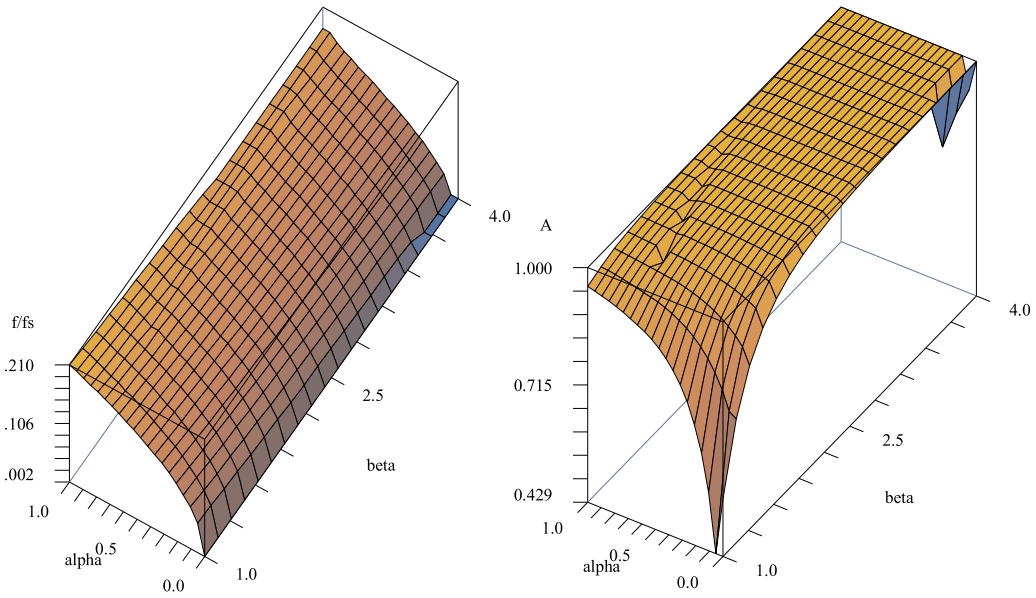

Abbildung 4.12: *Frequenz (links) und Amplitude (rechts) des Oszillators mit Integrator und Schmitt-Trigger in Abhängigkeit der beiden Parameter α und β.*

Per Definition der Wertebereiche von α und β ist $|\mathbf{W}_{IS}| > 1$ stets gegeben. Anders sieht es hingegen für die Bedingung aus Ungleichung 3.37 auf Seite 32 aus, welche komplex konjugierte Eigenwerte im Fixpunkt Null sichert – dies ist beim Oszillator mit Integrator und Schmitt-Trigger nicht mehr allgemein gegeben. Wie Abbildung 4.13 zeigt, schwingt der Oszillator beispielsweise für $\alpha = 0.01$ und $\beta = 2.4$ stabil.

Es gilt für diese Parameter aber:

$$\mathcal{D}(\mathbf{0}) = 2.41 \not> 2.873025 = \frac{1}{4}\mathcal{S}^2(\mathbf{0}). \tag{4.26}$$

Die Eigenwerte ergeben sich aus Gleichung 3.26 auf Seite 30 zu

$$\lambda_1 \approx 2.38, \tag{4.27}$$
$$\lambda_2 \approx 1.01. \tag{4.28}$$

Dies bedeutet, dass auf den Vektor im Phasenraum nahe Null keine Rotation wirkt. Doch wieso schwingt der Oszillator dann? Die Antwort findet man, wenn man den Parameter β geringfügig variiert. Für den Wert β_0 (siehe

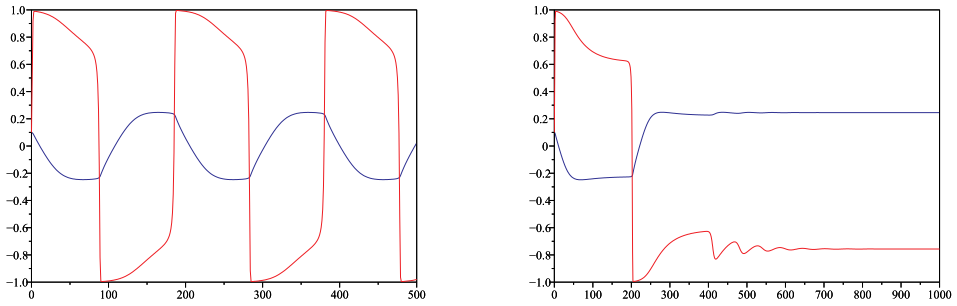

Abbildung 4.13: *Ausgangssignale des Oszillators mit Integrator und Schmitt-Trigger. (links)* $\alpha = 0.01$, $\beta = 2.4$. *(rechts)* $\beta = \beta_0 := 2.41314207820835347$.

Abbildung 4.13, rechts), kommt die Schwingung erstmalig zum Erliegen und man bemerkt einen stabilen Fixpunkt an der Stelle

$$\boldsymbol{x}_1^* \approx \begin{pmatrix} 0.245 \\ -0.757 \end{pmatrix}. \tag{4.29}$$

Aus Symmetriegründen ist dann auch $\boldsymbol{x}_2^* := -\boldsymbol{x}_1^*$ ein Fixpunkt. Für ein gewisses β_1, mit $(\beta_0 - 10^{-17}) < \beta_1 < \beta_0$, erfährt das System eine Bifurkation, bei der die beiden Fixpunkte $\boldsymbol{x}_{1,2}^*$ instabil werden. Das Phasendiagramm sowie die beiden Fixpunkte sieht man in Abbildung 4.14, oben rechts, in blau.

Bei den Fixpunkten handelt es sich um Sattelpunkte. Der größere Eigenwert beträgt ungefähr 1.4, während der andere knapp unter 1.0 liegt. Entsprechend sieht auch das Verhalten der Transiente aus. Man betrachte hierzu Abbildung 4.13, links. Die Transiente schleicht sich zunächst an den Fixpunkt heran (blaue Kurve) und entfernt sich dann rasch in eine andere Richtung (rote Kurve). Während die Annäherung durch den Eigenwert unter Eins gesteuert wird, ist für das Davonrennen der Eigenwert 1.4 verantwortlich. Wegen der geringen Auflösung sieht es in Abbildung 4.14 so aus, als verliefe die Transiente mitten durch die Fixpunkte. Dies ist selbstverständlich nicht der Fall – die Transiente kommt den Fixpunkten mit sinkender Oszillatorfrequenz jedoch beliebig nahe.

4.3 Vergleichende Übersicht der Oszillatoren

Zum Vergleich sind in Abbildung 4.14 die Phasendiagramme aller bisher vorgestellten Oszillatoren zusammengefasst. Neben der Sonderstellung des

Oszillators mit Integrator und Schmitt-Trigger fällt auf, dass für den großen Wert von $\epsilon = 0.99$ nur noch der SO(2)-Oszillator schwingt.

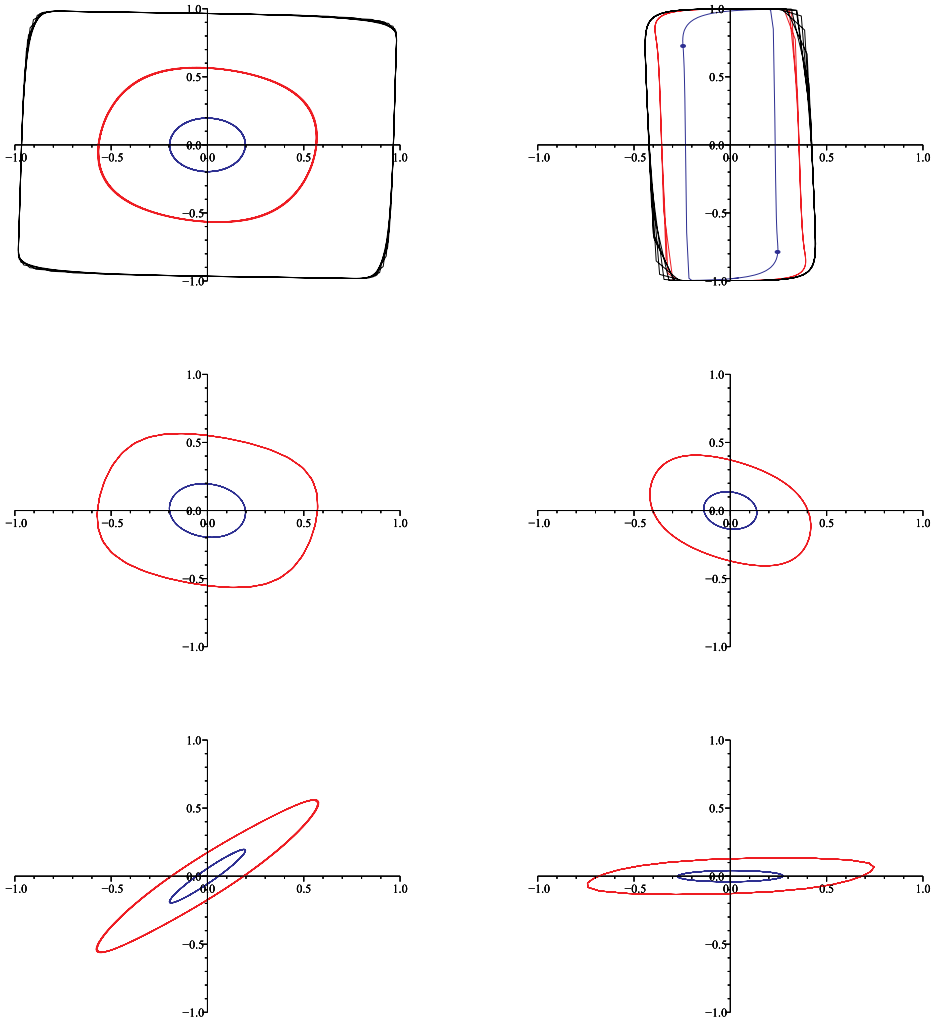

Abbildung 4.14: *Phasendiagramme der vorgestellten Oszillatoren, jeweils für* $\phi = 0.047$ *und* $\epsilon = 0.01$ *(blau),* $\epsilon = 0.1$ *(rot) und* $\epsilon = 0.99$ *(schwarz). Für den Oszillator mit Integrator und Schmitt-Trigger gelten die Werte* $\alpha = 0.01$, $\beta = 2.4$ *(blau),* $\alpha = 0.047$, $\beta = 3.5$ *(rot) und* $\alpha = 0.094$, $\beta = 3.5$ *(schwarz).*
Oben: (links) SO(2)-Oszillator. (rechts) Integrator mit Schmitt-Trigger.
Mitte: (links) Magic Circle. (rechts) Vereinfachter Magic Circle.
Unten: (links) Biquadratischer Oszillator. (rechts) Waveguide-Oszillator.

Wie man sieht, lassen sich schon bei zwei Neuronen und vier Verbindungsgewichten auf sehr unterschiedliche Weise Oszillatoren realisieren. Erhöht man die Anzahl der Neuronen, so steigt die Variationsvielfalt entsprechend an. Eine analytische Beschreibung solcher Systeme ist nur möglich, wenn homogene Verbindungsstrukturen vorliegen. Ein Beispiel hierfür liefert das nächste Kapitel.

Neuronale Netze mit 100 bis 1000 Neuronen und randomisierten Verbindungsgewichten wurden in [DMPS01] betrachtet. In Netzen dieser Größe kommt es prinzipiell stets zu Oszillationen und Synchronisationseffekten. Letztere wurden für gekoppelte Oszillatoren mit Schmitt-Trigger in [TS00] sowohl numerisch simuliert, wie auch in einer aufgebauten elektronischen Schaltung nachgewiesen.

Bemerkung zur Rechengenauigkeit

Bei den Analysen sämtlicher Oszillatoren wurde die übliche 80-Bit Fließkomma-Arithmetik verwendet, was schon alleine für die Berechnung der Lyapunov-Exponenten notwendig ist. Implementiert man jedoch einen der Oszillatoren auf einem Mikroprozessor, so wird man aus Effizienzgründen normalerweise in einem deutlich gröber gerasterten Zahlenraum agieren. Eine untere Grenze stellt hier die Verwendung von 8-Bit Festkommazahlen dar.

Abbildung 4.15 zeigt die Ausgangssignale des SO(2)-Oszillators für diesen Fall. Wie man sieht, sind die um $\pi/2$ phasenverschobenen Sinussignale noch deutlich erkennbar und genügen sicherlich auch, um einfache Laufmaschinen, wie beispielsweise den TED (siehe Kapitel 11.2), anzusteuern.

Bei reduzierter Rechengenauigkeit empfiehlt sich aufgrund des stabilen Schwingverhaltens die Verwendung des SO(2)-Oszillators. Aufgrund der erhöhten Parameterauflösung im unteren Frequenzbereich ist außerdem noch der Magic Circle-Oszillator in der vereinfachten Variante für Robotikanwendungen interessant.

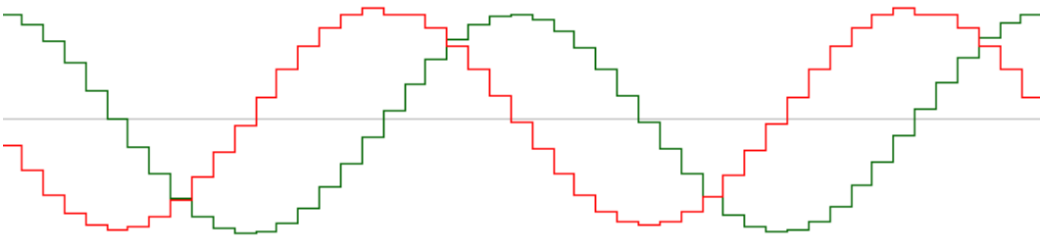

Abbildung 4.15: *Eine Sekunde Signalverlauf des SO(2)-Oszillators bei reduzierter 8-Bit-Rechengenauigkeit.*

Kapitel 5

Homöostatische Ringmodule für segmentierte Roboter

Wie bereits zu Anfang beschrieben, lassen sich alle phasenstarren Signale gleicher Grundfrequenz auf einfache Weise aus den beiden Ausgangssignalen eines einzigen, zentralen Quadraturoszillators ableiten. Die Verwendung eines solchen *Central Pattern Generator* (CPG) zählt zu den Standardmethoden, um die notwendigen motorischen Signale für Laufmaschinen mit vorgegebener Morphologie zu erzeugen.

Ein CPG für beliebige Signalverläufe kann auch mit Hilfe eines neuronalen Netzes implementiert werden (siehe [Ame03]), insbesondere zur Erzeugung der notwendigen Signale für vierbeinige Roboter, wie in [CBD+97] beschrieben.

Will man hingegen einen modular aufgebauten Roboter in Bewegung versetzen, dessen motorische Module auch zur Laufzeit noch umgesteckt werden können, dann ist unklar wo der CPG platziert sein sollte. Hier scheint sich eher eine Steuerungsstruktur anzubieten, die gleichmäßig über alle motorischen Module verteilt ist, so wie man dies auch bei segmentierten Organismen findet, beispielsweise bei Tausendfüßlern und Regenwürmern.

In diesem Kapitel wird ein Konzept für solche motorischen Neuromodule vorgestellt, die als n-dimensionale Verallgemeinerung des SO(2)-Oszillators verstanden werden können.

5.1 Oszillierende neuronale Ringe

Der in Kapitel 4.1 auf Seite 35 vorgestellte SO(2)-Oszillator kann auf naheliegende Weise zu einem Ringoszillator erweitert werden, indem weitere Neuronen eingefügt werden. Für $m \geq 2$ Neuronen erhält man folgende $m \times m$-

Gewichtsmatrix:

$$\mathbf{V}_m = (w_{i,j}) := \begin{pmatrix} s & r & & \\ & \ddots & \ddots & \\ & & s & r \\ -r & & & s \end{pmatrix}, \tag{5.1}$$

wobei s die Selbstkopplung und r die Ringkopplung der Neuronen darstellt, wie dies bereits bei der Gewichtsmatrix $\mathbf{W}_2$ des SO(2)-Oszillators beschrieben wurde. Außer den Selbst- und Ringkopplungen sind keine weiteren Verbindungen vorhanden, das heißt alle anderen Einträge in der Gewichtsmatrix sind Null.

Damit der Ringoszillator schwingt, muss notwendigerweise die Bedingung

$$\left(\prod_{i=2\ldots m} w_{i-1,i} \right) w_{m,1} < 0 \tag{5.2}$$

erfüllt sein. Aus Symmetriegründen ist hierbei die Position und Anzahl der Negierungen unerheblich, daher beschreibt für $m = 2n$, $n \geq 1$ auch die nachstehende Gewichtsmatrix, in welcher die Ringkopplungen alternierende Vorzeichen besitzen, einen schwingenden Ringoszillator:

$$\mathbf{U}_{2n} := \begin{pmatrix} s & r & & & \\ & s & -r & & \\ & & \ddots & \ddots & \\ & & & s & r \\ \beta r & & & & s \end{pmatrix}, \qquad \beta := (-1)^n. \tag{5.3}$$

Offensichtlich ist $\mathbf{W}_2 = \mathbf{V}_2 = \mathbf{U}_2$, allgemein gilt jedoch

$$\mathbf{V}_{2n} \neq \mathbf{U}_{2n}, \qquad \text{für } n > 1. \tag{5.4}$$

Ist n ungerade, dann hat man $\beta = -1$, so dass sich $\mathbf{U}_{2n}$ als Serienschaltung von n identischen Ringmodulen implementieren lässt. Abbildung 5.1 zeigt den Aufbau eines solchen Ringmoduls.

Das Modul hat an jedem seiner zwei Enden einen Schalter, der die beiden Anschlüsse überbrückt, solange kein anderes Modul eingesteckt ist. Auf diese Weise funktioniert ein einzelnes Modul bereits als SO(2)-Oszillator. Steckt man mehrere Module zusammen, so öffnen sich die entsprechenden Schalter und die einzelnen Ringe vereinen sich zu einem einzigen großen. Bei einer geraden Anzahl von Modulen muss an einer beliebigen Stelle eine zusätzliche Negierung eingefügt werden, was auf verschiedene Weise geschehen kann.

Entweder erhält jedes Modul einen zusätzlichen Schalter, mit dem zwischen $\pm r$ hin- und hergeschaltet werden kann. Dieser Schalter kann dann zugleich als Ein-/Ausschalter der Oszillation fungieren. Oder man sieht eine weitere, nicht-negierende Steckverbindung pro Modul vor.

Sieht man an jedem Modul ein Gelenk vor, dessen motorischer Antrieb aus der Linearkombination der beiden internen Oszillatorsignale generiert wird, dann lassen sich interessante Laufmaschinen zusammenstecken. Dreht

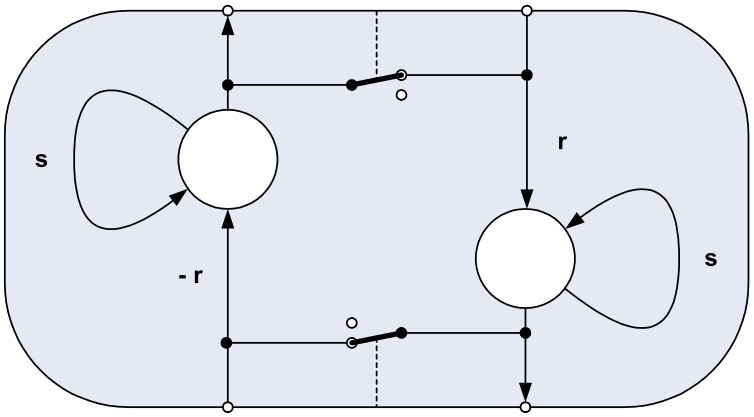

Abbildung 5.1: *Aufbau eines einzelnen Ringmoduls, das mit anderen seiner Art zusammengesteckt werden kann.*

man ein Modul um 180°, dann ändert sich die Phasenlage aller Signale, so dass ein neues Bewegungsverhalten entsteht.

Berechnung der Eigenwerte

Aufgrund der einfachen Struktur von $\mathbf{U}_{2n}$ lassen sich die Eigenwerte rasch bestimmen, indem man die Determinante von $\mathbf{U}_{2n} - \lambda\mathbf{I}$ gemäß Laplace nach der ersten Spalte entwickelt.

$$|\mathbf{U}_{2n} - \lambda\mathbf{I}| \stackrel{!}{=} 0, \tag{5.5}$$

$$(s-\lambda)^{2n} + r^{2n} = 0, \tag{5.6}$$

$$(s-\lambda)^{2n} = -r^{2n}. \tag{5.7}$$

Man substituiert

$$(s-\lambda) = \varrho \mathrm{e}^{-\mathrm{i}\pi\psi} \tag{5.8}$$

und erhält weiter

$$\varrho^{2n}\mathrm{e}^{-\mathrm{i}\pi\cdot 2n\psi} = -r^{2n}. \tag{5.9}$$

Durch Koeffizientenvergleich schließt man

$$\varrho = r \tag{5.10}$$

und

$$\mathrm{e}^{-\mathrm{i}\pi\cdot 2n\psi} = -1, \tag{5.11}$$

$$2n\psi = 2k+1, \qquad k = 0, 1, \ldots \tag{5.12}$$

$$\psi = \frac{k+1/2}{n}. \tag{5.13}$$

Löst man die Substitutionsgleichung 5.8 nach λ auf und setzt die Werte für ϱ und ψ ein, so ergibt sich schließlich die folgende Menge von komplex konjugierten Eigenwerten:

$$\lambda_k = s - r\mathrm{e}^{-i\pi\frac{k+1/2}{n}}, \qquad k = 0, 1, \ldots, 2n-1. \tag{5.14}$$

Wie in Abbildung 5.2 zu sehen ist, liegen die Eigenwerte in der komplexen Zahlenebene auf einem Kreis um den Mittelpunkt $M(s, 0)$ mit Radius r.

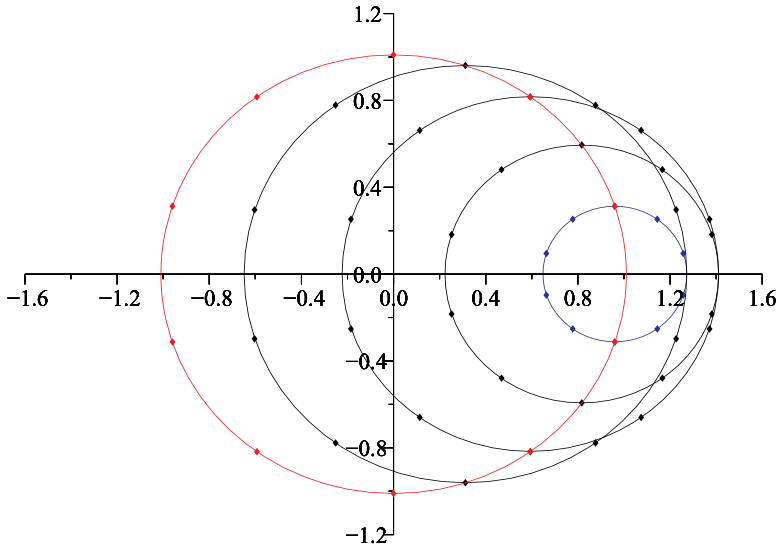

Abbildung 5.2: *Eigenwerte des 10-Rings in der komplexen Zahlenebene für* $\epsilon = 0.01$, $\phi = 0.05$ *(blau),* $\phi = 0.10 \ldots 0.20$ *(schwarz) und* $\phi = 0.25$ *(rot). Die Werte von* s *und* r *sind hierbei durch die Gleichungen 5.15 und 5.16 festgelegt.*

Bei n zusammengesteckten Ringmodulen liegt ein $2n$-Neuronen-Ring vor, der insgesamt n Paare komplex konjugierter Eigenwerte besitzt. Wie man unmittelbar erkennt, sind die Eigenwerte genau dann echt komplex, wenn $r > 0$, also eine Ringkopplung vorhanden ist. In [Lai98] findet man eine Diskussion der Eigenwerte bei kontinuierlichem Zeitmodell.

Frequenz und Amplitude

Definiert man analog zum SO(2)-Oszillator

$$s := (1 + \epsilon) \cdot \cos(2\pi\phi), \tag{5.15}$$

$$r := (1 + \epsilon) \cdot \sin(2\pi\phi) \tag{5.16}$$

und berechnet unter Verwendung der Gewichtsmatrix $\mathbf{U}_{2n}$ die Schwingungsfrequenz von $n = 3$ beziehungsweise $n = 5$ zusammengesteckten Ringmodulen, dann erhält man die in Abbildung 5.3 gezeigten Verläufe.

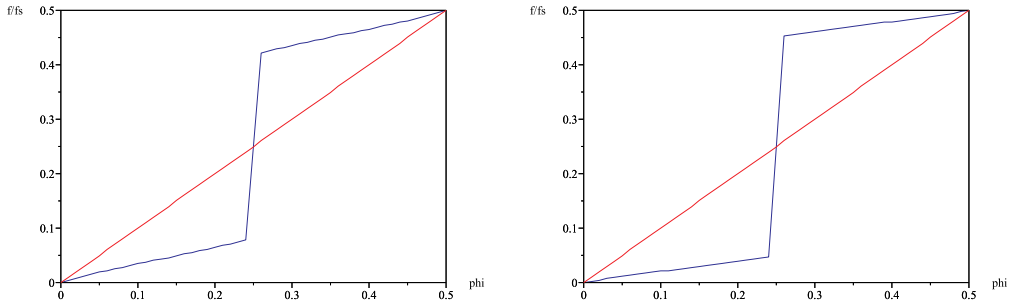

Abbildung 5.3: *(links) Relative Frequenz f/f_s von drei zusammengesteckten Ringmodulen in Abhängigkeit von ϕ, für $\epsilon = 0.01$. Neben der stabilen Grundfrequenz (blau) kann auch die zweite Oberwelle (rot) als Schwingfrequenz dienen. (rechts) Grundfrequenz (blau) und vierte Oberwelle (rot) bei fünf zusammengesteckten Ringmodulen.*

Je nach Anfangswert landet man in einem der koexistierenden quasiperiodischen Orbits unterschiedlicher Frequenz. Das Einzugsgebiet des zur relativen Schwingungsfrequenz $f_{max} = \phi$ gehörigen Orbits wird mit zunehmender Neuronenzahl des Rings kleiner. Bei einem 10-Neuronen-Ring genügt eine Abweichung in der Größenordnung von 10^{-3}, um den Orbit zu verlassen und auf den Attraktor der Grundfrequenz f_0 des Oszillators zu fallen. Für n zusammengesteckte Ringmodule beträgt die Grundfrequenz näherungsweise

$$f_0 \approx \frac{\phi}{n}, \qquad 0 < \phi < \frac{1}{4}. \tag{5.17}$$

Ist $k > 1$ ein Teiler von $2n$, dann existiert wegen der inneren Symmetrie des Rings immer auch ein Orbit mit der Frequenz

$$f_k \approx \frac{k}{2n} \cdot \phi. \tag{5.18}$$

Bei drei Ringmodulen sind demnach die Frequenzen $\phi/3$, $\phi/2$ und ϕ möglich, bei fünf Ringmodulen die Frequenzen $\phi/5$, $\phi/2$ und ϕ. Von praktischem Interesse ist jeweils nur der tieffrequente Attraktor, welcher das größte Einzugsgebiet besitzt. In [GS93], [GSBC98], [GPS03] und [GST05] findet man weitergehende gruppentheoretische Symmetriebetrachtungen zur Bifurkation gekoppelter Module. Eine systematische Analyse aller nicht-quasiperiodischen Attraktoren wurde in [Pas95], [Pas98] und [Pas02] vorgenommen.

Erhöht man bei konstantem ϕ die Anzahl der Ringmodule, sinkt die Frequenz entsprechend ab. Gleichzeitig steigt die Amplitude an, da mehr Selbstkopplungen vorhanden sind, welche die Energie des Gesamtsystems erhöhen. Ist die gewünschte Frequenz des Ringoszillators gering, wie beispielsweise bei der Steuerung von Laufmaschinen, dann gilt vereinfachend

$$s = (1 + \epsilon) \cdot \cos(2\pi\phi) \approx 1, \tag{5.19}$$

$$r = (1 + \epsilon) \cdot \sin(2\pi\phi) \approx 2\pi\phi. \tag{5.20}$$

Die Selbstkopplungsgewichte sind also stets nahe Eins. Sie steuern die im System zirkulierende Energiemenge und somit die Signalamplitude. Auf der anderen Seite sind die Ringkopplungen proportional zur Frequenz, wobei der Proportionalitätsfaktor von der Anzahl der Neuronen abhängt.

Mit Hilfe dieser Zusammenhänge lassen sich Ringmodule entwerfen, die ihre Gewichte derart selbst regulieren, dass Frequenz und Amplitude vorbestimmte Werte beibehalten – auch wenn Module hinzugefügt oder entfernt werden.

5.2 Herleitung homöostatischer Lernregeln

Ein zentraler Grundsatz der modernen Physiologie ist die Homöostase. Sie beschreibt das Prinzip, dass jeder lebende Organismus nach einer Störung versucht, sich wieder in einen gesunden internen Gleichgewichtszustand einzupendeln.

Die Dynamik eines neuronalen Netzes bestimmt sich aus der Stärke der Verbindungsgewichte, daher liegt es nahe, Regelungsprozesse über Gewichtsveränderungen zu realisieren. Auf welche Weise diese Veränderungen geschehen bestimmt im Einzelfall die sogenannte Lernregel. Je nach intendierter Anwendung existieren verschiedene Lernregeln.

Im Bereich Robotik gibt es beispielsweise Lernregeln, die adaptives Verhalten produzieren, so dass die damit ausgestatteten Roboter in unterschiedlichen Umwelten agieren können (siehe [UF01] und [BF02]). Einige Lernregeln sind jedoch auch in der Lage, exploratives Verhalten hervorzubringen, so zum Beispiel die in [DP99] definierte. Wie auch die weiter unten eingeführten Lernregeln, wurde sie ebenfalls vom Prinzip der Homöostase abgeleitet.

Man kann nun definieren, dass das oben eingeführte Ringmodul seinen Gleichgewichtszustand erreicht hat, wenn es mit einer vorher festgelegten Zielfrequenz f_0 und Zielamplitude A_0 schwingt. Die Phasenverschiebung ist aus der Gleichgewichtsdefinition ausgeklammert, da sie über die Art und Weise des Zusammensteckens variiert werden kann.

Bevor man homöostatische Regelungsvorgänge für das Ringmodul entwickeln kann, muss zunächst ein geeignetes Maß für die Erreichung des Gleichgewichtszustands gefunden werden. Hierzu tastet man die Periode einer Sinusschwingung zu N diskreten, äquidistanten Zeitpunkten ab und betrachtet die beiden folgenden Summen:

$$S_A(A,N) := \frac{1}{N}\sum_{k=0}^{N-1}\left|A\sin\left(\frac{2\pi k}{N}\right)\right|, \tag{5.21}$$

$$S_f(A,N) := \frac{1}{N}\sum_{k=0}^{N-1}\left|A\sin\left(\frac{2\pi(k+1)}{N}\right) - A\sin\left(\frac{2\pi k}{N}\right)\right|. \tag{5.22}$$

Die erste Summe $S_A(A,N)$ berechnet die durchschnittliche Amplitude des gleichgerichteten Signals und kann auf einfache Weise wie folgt approximiert werden:

$$S_A(A,N) \approx \lim_{N\to\infty} S_A(A,N), \tag{5.23}$$

$$= \frac{1}{2\pi}\int_0^{2\pi} |A\sin x|\,dx, \tag{5.24}$$

$$= \frac{2}{\pi}\cdot A. \tag{5.25}$$

Eine Sinusschwingung mit der Spitzenamplitude A bewirkt demnach ein durchschnittliches absolutes neuronales Ausgangssignal von $2A/\pi$, welches von der Frequenz unabhängig ist.

Für $N > 3$ lässt sich die zweite Summe $S_f(A, N)$ durch

$$S_f(A,N) \approx \frac{4}{N} \sum_{k=0}^{\lfloor N/4 \rfloor - 1} \left| A \sin\left(\frac{2\pi(k+1)}{N}\right) - A \sin\left(\frac{2\pi k}{N}\right) \right|, \tag{5.26}$$

$$= \frac{4A}{N} \sin\left(\frac{2\pi \lfloor N/4 \rfloor}{N}\right), \tag{5.27}$$

$$\approx \frac{4A}{N} \tag{5.28}$$

approximieren. Der Fehler $E_f(A, N) = 4A/N - S_f(A, N) \geq 0$ ist Null, wenn N ein Vielfaches von Vier ist. Allgemein liegt der relative Fehler für $N > 10$ unter 3% und für $N > 18$ unter 1%.

Nimmt man eine konstante Amplitude an, dann ist die durchschnittliche Veränderung des neuronalen Ausgangssignals umgekehrt proportional zur Schwingungsfrequenz. Über die durchschnittliche Aktivität und die durchschnittliche Aktivitätsveränderung am Einzelneuron lässt sich also messen, ob der Ring mit der gewünschten Frequenz und Amplitude schwingt und somit seinen Gleichgewichtszustand erreicht hat.

Lernregeln

Nachdem ein geeignetes Maß für den Gleichgewichtszustand gefunden ist, benötigt man ein Reihe von Beispielwerten stabil schwingender Ringe mit unterschiedlicher Neuronenanzahl. Ausgehend von einer servogetriebenen Laufmaschine setzt man die Berechnungsfrequenz der Ringmodule auf $f_s = 50\text{Hz}$ und die Zielfrequenz bei moderater Laufgeschwindigkeit auf $f_0 = 1\text{Hz}$. Damit ergibt sich $\phi = f_0/f_s = 1/50$.

Bei wachsender Amplitude nehmen wegen der Nichtlinearität des Tangens Hyperbolicus die Verzerrungen zu (siehe Abbildung 2.4 auf Seite 15). Mit der Zielamplitude $A_0 = 0.4$ wählt man jedoch einen guten Kompromiss zwischen hoher Amplitude und geringen Verzerrungen. Durch numerische Simulation von $n \in \{1, 2, \ldots, 6\}$ zusammengesteckten Ringmodulen erhält man bei der genannten Zielfrequenz und -amplitude die Gewichtswerte in Tabelle 5.1.

Startet man mit drei gekoppelten Ringmodulen und fügt ein weiteres hinzu, dann steigt die Amplitude um $\Delta A = 0.2$ von 0.4 auf 0.6 an. Um diesen Anstieg auszugleichen, muss die Selbstkopplung um $s_8 - s_6 = -0.093$ korrigiert werden. Entfernt man hingegen ein Ringmodul, muss die Selbstkopplung entsprechend um $s_4 - s_6 = +0.091$ korrigiert werden. Der mittlere absolute Korrekturwert beträgt somit $\Delta s = 0.092$.

Kombiniert man die numerisch ermittelten Daten mit der Abschätzung von $S_A(A, N)$ aus Gleichung 5.23, dann erhält man die folgende amplituden-

n	$2n$	s_{2n}	r_{2n}	$s_{2n} + r_{2n}$
1	2	1.036	0.132	1.168
2	4	0.905	0.187	1.092
3	6	0.814	0.260	1.074
4	8	0.721	0.345	1.066
5	10	0.642	0.420	1.062
6	12	0.550	0.510	1.060

Tabelle 5.1: *Gewichtswerte für $f_s = 50Hz$, $f_0 = 1Hz$ und $A_0 = 0.4$ bei n gekoppelten Ringmodulen, beziehungsweise $2n$ Neuronen.*

stabilisierende Lernregel für die Selbstkopplungsgewichte:

$$s(t+1) := s(t) + s_\delta(t), \tag{5.29}$$

$$s_\delta(t) := \varepsilon_s \frac{\Delta s f_0}{\Delta A f_s} \left(\frac{2}{\pi} A_0 - |x(t)| \right), \tag{5.30}$$

wobei $s(t)$ das Gewicht und $x(t)$ das Ausgangssignal des Neurons zum Zeitpunkt t darstellt. A_0 und f_0 sind die Zielamplitude und -frequenz, Δs, ΔA und f_s sind wie weiter oben definiert. ε_s stellt die normalisierte Lernrate dar, das heißt $\varepsilon_s = 1$ bedeutet, dass nach Hinzufügen oder Entfernen eines Ringmoduls die Zielamplitude A_0 zum Ende der nächsten vollen Schwingungsperiode wieder eingeregelt ist.

An dieser Stelle sind zwei Dinge zu bemerken: Erstens dient der Faktor f_0 in der Lernregel nur der Normalisierung von ε_s und ist für den Ablauf des homöostatischen Regelungsprozesses irrelevant.

Zweitens sind alle Parameter von vornherein bekannt, so dass die Lernregel in der einfachen Form

$$s_\delta(t) = \eta \left(\mu - |x(t)| \right) \tag{5.31}$$

mit vorausberechnetem η und μ implementiert werden kann. Da nur eine Multiplikation, eine Addition und eine Vorzeichenumkehr benötigt werden, lässt sich die Lernregel sogar auf langsameren 8-Bit-Mikroprozessoren einsetzen.

Auf analoge Weise erhält man die frequenzstabilisierende Lernregel für die Ringkopplungsgewichte:

$$r(t+1) := r(t) + r_\delta(t), \tag{5.32}$$

$$r_\delta(t) := \varepsilon_r \frac{\Delta r}{\Delta S_f f_s} f_0 \left(\frac{4A_0}{f_s} f_0 - |x(t) - \hat{x}(t)| \right), \tag{5.33}$$

wobei $r(t)$ das Gewicht zum Zeitpunkt t darstellt. $x(t)$ und $\hat{x}(t)$ sind die Ausgangssignale der Neuronen nach und vor der $r(t)$-gewichteten Verbindung, ε_r ist die normalisierte Lernrate und die übrigen Symbole sind wie weiter oben definiert.

Auch diese Lernregel kann wieder in der einfachen Form

$$r_\delta(t) = \eta\,(\mu - |x(t) - \hat{x}(t)|) \tag{5.34}$$

mit vorausberechnetem η und μ geschrieben werden. Man bemerke, dass die Gleichungen bis auf den Term $-\hat{x}(t)$ identisch sind. Offensichtlich kann der Term $\hat{x}(t)$ in der ersten Lernregel nur Null sein, da bei einer Selbstkopplung kein zweites Neuron involviert ist.

Da $r_\delta(t)$ die Schwingungsfrequenz steuert, aber gleichzeitig von A_0 und f_0 abhängt, funktioniert die Lernregel nur dann korrekt, wenn sie zusammen mit der Lernregel für die Selbstkopplungen eingesetzt wird. Hierbei dürfen die Lernraten ε_s und ε_r nicht zu hoch gewählt werden, sonst werden die Sinusschwingungen deformiert und es kommt zu unkontrollierbaren Interaktionen zwischen den beiden Lernregeln.

Diffusionsprozesse

Steckt man ein paar Ringmodule zusammen, die aufeinander abgestimmte Parameter besitzen, und schaltet sie ein, dann korrigieren die beiden Lernregeln Frequenz und Amplitude zuverlässig innerhalb weniger Schwingungsperioden. Dies funktioniert für ein einzelnes Modul genauso gut wie für sechs und mehr Module.

Trennt man jedoch einige Module und steckt sie im eingeschalteten Zustand wieder beliebig zusammen, so kann unter Umständen folgendes passieren. Da die getrennten Module weiterschwingen und unmittelbar beginnen, ihre Gewichte voneinander unabhängig zu verändern, liegt zum Zeitpunkt des Zusammensteckens eine unvorhersehbare Verteilung von Gewichtswerten und neuronalen Aktivitäten innerhalb der Module vor.

In ungünstigen Fällen stellen sich dann ungleiche Gewichtsverteilungen im zusammengesteckten Ring ein. Frequenz und Amplitude werden von den Lernregeln zwar immer noch auf den Zielwerten gehalten, aber die relative Phasenverschiebung zwischen einzelnen Modulen sowie zwischen den beiden Neuronen innerhalb eines Moduls sind nicht mehr gleich.

Um dies zu kompensieren, kann man zwei unabhängige Diffusionsprozesse innerhalb des Moduls vorsehen – einen für das Gewichtspaar s, s' und einen

für r, r':

$$\bar{s}(t) := \frac{s(t) + s'(t)}{2}, \tag{5.35}$$

$$s(t+1) := \delta_s \bar{s}(t) + (1 - \delta_s)\, s(t), \tag{5.36}$$

$$s'(t+1) := \delta_s \bar{s}(t) + (1 - \delta_s)\, s'(t) \tag{5.37}$$

und entsprechend für die Ringkopplungen:

$$\bar{r}(t) := \frac{r(t) + r'(t)}{2}, \tag{5.38}$$

$$r(t+1) := \delta_r \bar{r}(t) + (1 - \delta_r)\, r(t), \tag{5.39}$$

$$r'(t+1) := \delta_r \bar{r}(t) + (1 - \delta_r)\, r'(t). \tag{5.40}$$

Die Diffusionsprozesse werden unmittelbar nach den Gewichtsaktualisierungen der Lernregeln angewandt. Es ist ausreichend, für δ_s und δ_r kleine Werte zu nehmen, also $0 < \delta_s, \delta_r \ll 1$.

Implementiert man die Diffusionsprozesse zusammen mit den Lernregeln auf einem 8-Bit-Mikroprozessor, so werden nur Additionen benötigt, denn eine Division durch Zwei kann durch eine Rechtsverschiebung umgesetzt werden. Analog gilt dies, wenn man für δ_s und δ_r Zweierpotenzen vorsieht.

Gewichtsbegrenzung

Wie schon weiter oben beschrieben, ist es praktisch die Ringmodule mit einem Schalter auszustatten, der eine zusätzliche Negierung in den Signalfluss einfügt, wie dies bei einer geraden Anzahl zusammengesteckter Module notwendig ist. Mit diesem Schalter kann zugleich die Oszillation des gesamten Rings ausgeschaltet werden, so dass sich die Ausgangssignale in einer konstanten Aktivitätsverteilung einpendeln. Je nach dem an welchen Modulen die zusätzliche Negation aktiviert ist, ergeben sich unterschiedliche Aktivitätsverteilungen.

Der Nachteil dieser Start-Stop-Variante besteht darin, dass die Gewichte während der Stop-Phasen von den Lernregeln über alle Grenzen gefahren werden. Dies lässt sich jedoch auf einfache Weise eindämmen.

Wie man der Tabelle 5.1 auf Seite 57 entnehmen kann, liegt die Summe $s_{2n} + r_{2n}$ bei n Ringmodulen stets leicht über Eins. Der maximale Wert wird beim Einzelmodul mit 1.168 erreicht. Vergleicht man die Absolutsumme der beiden Gewichte eines Neurons mit diesem Wert und setzt die Gewichte bei Überschreitung entsprechend herab, dann schwingen sich zusammengesteckte Ringmodule nach dem Einschalten stets sofort wieder ein.

Arbeitet das System nicht mit Fließkommazahlen, sondern mit stark begrenzter Festkomma-Arithmetik, wie beispielsweise beim TED (siehe Kapitel 11.2), dann kann man alternativ den Negationsschalter und die Gewichtsbegrenzung auch wegfallen lassen: falls die Schwingung aufgrund einer fehlenden Negation zum Erliegen kommt, wachsen die Ringkopplungen über alle Grenzen und es kommt zum Zahlenüberlauf. Sobald das erste Ringgewicht durch den Überlauf sein Vorzeichen wechselt, beginnt der Ring zu schwingen und der homöostatische Gleichgewichtszustand kann sich wieder einstellen.

5.3 Ergebnisse der Funktionstests

Die Wirkungsweise der homöostatischen Lernregeln und Diffusionsprozesse lässt sich illustrieren, indem man das Zusammenstecken und Auseinanderziehen der Ringmodule zunächst mit festen Gewichten, dann mit homöostatischer Gewichtsregelung bei laufendem Betrieb simuliert.

Bei der ersten Simulation wurden die Selbstkopplungsgewichte fest auf $s = 0.814$ und die Ringkopplungsgewichte auf $r = 0.260$ gesetzt. Die Simulation wurde mit drei zusammengesteckten Ringmodulen gestartet. Das Ergebnis ist in Abbildung 5.4 zu sehen.

Wie erwartet schwingen die Ringmodule mit der Amplitude $A_0 = 0.4$ und dem Frequenzverhältnis $f_0/f_s = 1/50$, was bei 50 Zeitschritten pro Sekunde einer Ausgangsfrequenz von 1.0Hz entspricht.

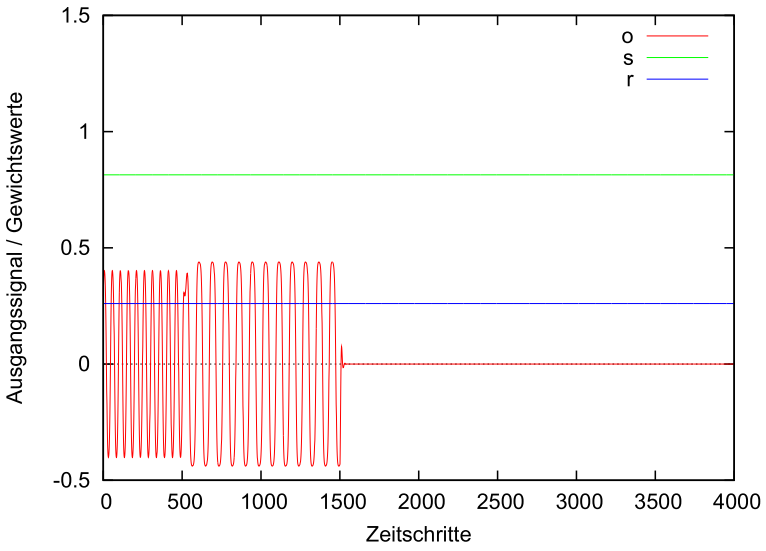

Abbildung 5.4: *Neuronales Signal am Ausgang eines Ringmoduls (rot) bei Verwendung von festen Gewichten für Selbstkopplung (grün) und Ringkopplung (blau).*

Nach 10 Sekunden (zum Zeitschritt 500) werden zwei zusätzliche Ringmodule eingesteckt, wodurch die Amplitude leicht ansteigt und die Frequenz deutlich von 1.0Hz auf 0.6Hz absinkt. Weitere 20 Sekunden später (zum Zeitschritt 1500) werden vier der fünf Ringmodule entfernt und fast unmittelbar darauf bricht die Schwingung vollständig zusammen. Erneut 20 Sekunden später (zum Zeitschritt 2500) werden zwei Ringmodule hinzugefügt, so dass die Anfangskonfiguration wiederhergestellt ist. Das System ist jedoch nicht in der Lage sich aus der Nulllage zu erholen. Theoretisch sollte die Schwingung wieder einsetzen, doch bedingt durch die endliche Genauigkeit der numerischen Berechnung ist dies praktisch nie der Fall – selbst bei Verwendung von 80-Bit-Fließkommazahlen.

Der vollständige Ablauf des Auseinander- und Zusammensteckens wird nun unter Verwendung der homöostatischen Regelprozesse wiederholt. Um vergleichbare Bedingungen zu haben, werden entsprechende Parameterwerte für Zielamplitude, Zielfrequenz und Berechnungsfrequenz der Module verwendet. Wie weiter oben bereits beschrieben, ist es entscheidend, zunächst eine stabile Amplitude zu erreichen, da die frequenzregelnde Veränderung der Ringkopplungsgewichte von der Amplitude abhängt. Die amplitudenstabilisierende Veränderung der Selbstkopplungsgewichte ist hingegen frequenzunabhängig. In Tabelle 5.2 auf Seite 63 sind Parameterwerte aufgelistet, welche sich als robust erwiesen haben.

Unter Verwendung dieser Werte verhalten sich die Ringmodule wie in Abbildung 5.5 gezeigt. Offensichtlich finden die Ringmodule wieder zurück zu ihrem Gleichgewichtszustand, nachdem sie durch das Umstecken im laufenden Betrieb in ihrer Schwingung gestört wurden. Wie man sieht, erreicht nach wenigen Perioden jeweils zuerst die Amplitude mit 0.4 wieder ihren definierten Zielwert. Danach pendelt sich etwas langsamer die Frequenz ein (siehe Abbildung 5.5, Zeitschritte 500 bis 700, 1600 bis 2000 und 2500 bis 3000).

Zusammenfassend lassen sich im Vergleich zur Simulation mit festen Gewichten folgende drei Tatsachen feststellen:

- Je nach Anzahl der aktiven Ringmodule streben alle Gewichte ihren optimalen Wert an (siehe Tabelle 5.2). Die Startgewichte $s = 0.814$ und $r = 0.260$ werden wieder erreicht, sobald die Anzahl der Ringmodule dieselbe wie in der Anfangskonfiguration ist. Das bedeutet, dass letztlich jedes Ringmodul in der Lage ist, die Anzahl der angeschlossenen Ringmodule allein aus den eigenen Gewichten zu schließen, ohne dass eine spezielle Kommunikation hierfür notwendig wäre. Es spielt dabei auch keine Rolle, wie weit die anderen Module entfernt sind. Lax gesagt

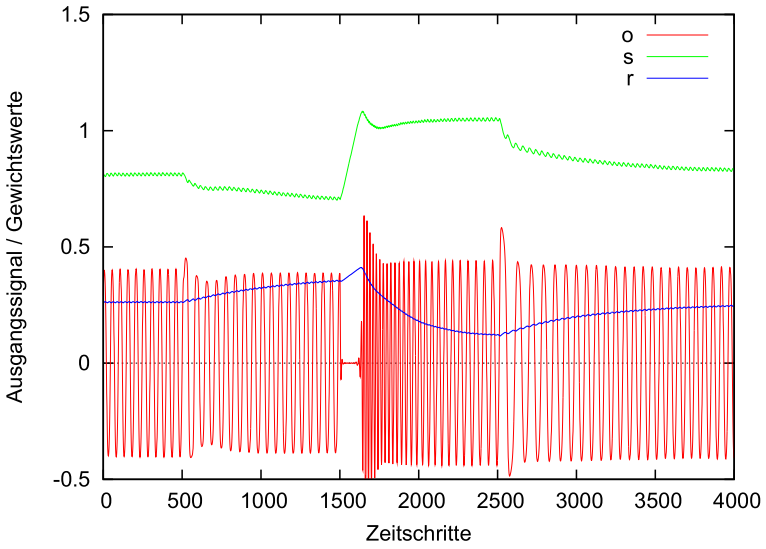

Abbildung 5.5: *Neuronales Ausgangssignal (rot) bei Verwendung der homöostatischen Regelprozesse für Selbstkopplung (grün) und Ringkopplung (blau).*

kann jedes Ringmodul den Rest des künstlichen Organismus fühlen und morphologische Veränderungen spüren.

- Die Schwingungen brechen niemals zusammen (vergleiche images 5.4 und 5.5, Zeitschritte 1500 bis 1700). In Analogie zur Physiologie könnte man sagen, dass die Homöostase den künstlichen Organismus vor dem Tod bewahrt.

- Bei genauer Betrachtung der sich langsam verändernden Selbstkopplungsgewichte zeigt sich eine leichte Modulation mit der Zielfrequenz (man sieht dies am besten in Abbildung 5.5 kurz nach Zeitschritt 2500). Dies bedeutet, dass die Lernrate der Selbstkopplungsgewichte optimal gewählt wurde. Eine höhere Lernrate würde zur Destabilisierung des Regelprozesses führen, während eine niedrigere Lernrate die Reaktionszeit unnötig verlängern würde.

Nach diesen analytischen und eher philosophischen Überlegungen sei noch angemerkt, dass sich die homöostatischen Ringmodule auch praktisch aufbauen lassen und zur motorischen Ansteuerung künstlicher Kreaturen eignen. Ein Baukastensystem aus aktuierten, modularen Einzelmodulen, wie es in [Raf04] beschrieben wird, wäre eine adäquate Anwendung. Über einen zusätzlichen Schalter, der innerhalb eines Moduls die Ringkopplung negiert, ließe sich das gesamte System zwischen Schwingungserzeugung und tonischen

neuronalen Ausgangssignalen umschalten. In Abhängigkeit der Schalterstellungen aller Module ergeben sich dann unterschiedliche Ruhepositionen.

Parameter	Symbol	Wert
Zielamplitude	A_0	0.4
Zielfrequenz	f_0	1Hz
Berechnungsfrequenz	f_s	50Hz
Lernrate SK	ε_s	1.0
Lernrate RK	ε_r	5.0
Diffusionsrate SK	δ_s	0.05
Diffusionsrate RK	δ_r	0.1

Tabelle 5.2: *Optimale Parameterwerte für Konfigurationen mit bis zu sechs Ringmodulen. Selbst- und Ringkopplungsgewichte sind mit SK und RK abgekürzt.*

Kapitel 6

Monostabile Neuromodule für komplexe Bewegungsabläufe

Die in den vorangegangenen Kapiteln vorgestellten Oszillatoren sind vor allem geeignet, um zyklische Bewegungsabläufe zu steuern. Viele Aufgabenstellungen erforden jedoch Folgen von komplexen Bewegungen, die ineinander übergehen. Dies gilt insbesondere für humanoide Roboter, wie beispielsweise die in Kapitel 11.5 vorgestellten. Eine weit verbreitete Technik zur Ansteuerung solcher Bewegungssequenzen ist die Verwendung sogenannter *Keyframes*. Ein Keyframe repräsentiert hierbei eine bestimmte Körperpose und Interpolation zwischen unterschiedlichen Keyframes versetzt den Roboter in Bewegung.

Die Verwendung von Keyframes hat diverse Vorteile. So lassen sich auf einfache Weise mit dem Roboter Körperposen formen und abspeichern, wenn dieser die Gelenke antriebsfrei geschaltet hat. Die so gewonnenen Keyframes kann man mit unterschiedlichen Zeitverzögerungen abspielen und optimieren, bis eine gewünschte Bewegungsfolge erreicht ist.

Andererseits wächst eine umfassende Bibliothek von Bewegungsdaten für humanoide Roboter mit vielen Freiheitsgraden, wie die in Abbildung 6.1 gezeigte, rasch an. Es wird dann unhandlich, mit einer solchen Bibliothek zu arbeiten – sie auf unterschiedliche Roboter einer Bauserie anzupassen oder nach mechanischen Wartungsarbeiten alle Bewegungen zu überprüfen. Wenn auf den Mikrocontrollern der Zielplattform nicht genügend Speicher vorhanden ist, müssen die Bewegungsdaten außerdem in einer komprimierten Form abgespeichert werden, zum Beispiel mit Hilfe der in [ITN03] vorgestellten Methode, die versucht, automatisch ein Modell für die abzuspeichernden Bewegungsdaten zu konstruieren.

Neben einer kompakten Beschreibung von Bewegungssequenzen ist die geeignete Stabilisierung der Bewegungen genauso wünschenswert. Es ist al-

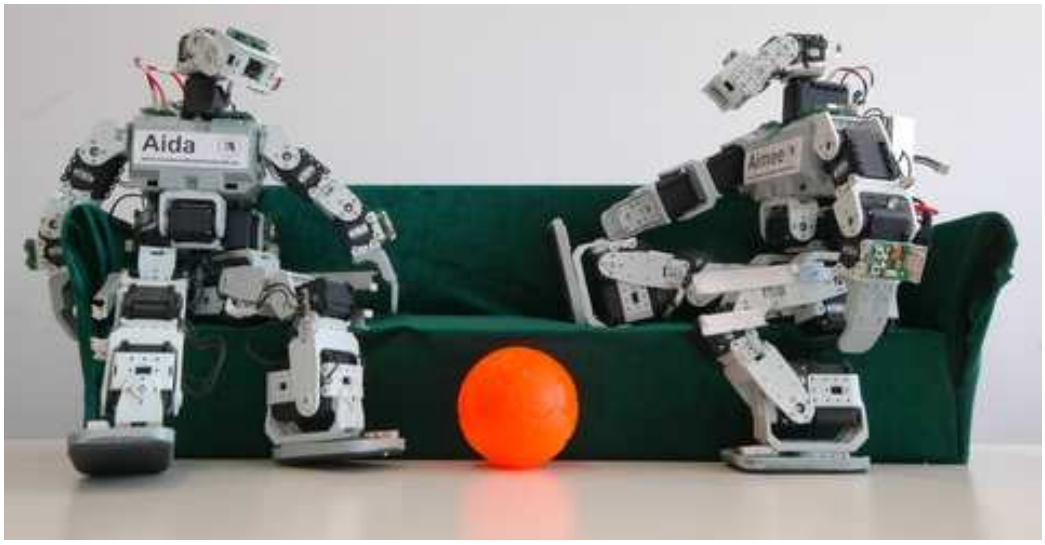

Abbildung 6.1: *Die humanoiden Roboter Aida und Aimee vom Humanoid Team Humboldt können komplexe Körperbewegungen durchführen. Jeder der beiden Roboter besitzt 21 Freiheitsgrade. Eine detaillierte Beschreibung findet man Kapitel 11.5.*

lerdings nicht unmittelbar klar, wie die zur Stabilisierung notwendigen sensorischen Rückkopplungen im Rahmen der Keyframe-Technik berücksichtigt werden können. Sensomotorische Schleifen werden in der Regel neuronal realisiert (siehe [NAA03] und [RVSA06]).

Prinzipiell sind neuronale Attraktoren geeignet, um diskrete Bewegungssequenzen beliebiger Länge zu produzieren (siehe [Pas95]), oder auch fließende Bewegungen unter Verwendung quasiperiodischer Attraktoren, wie in Kapitel 4 und [PHZ03] sowie [HP07] beschrieben. Es ist jedoch nicht trivial, rekurrente neuronale Netze mit vorgeschriebener Attraktorlandschaft zu finden, wenn gleichzeitig explizite Zeitverläufe eingehalten werden sollen. Im folgenden wird vorgestellt, wie man eine existierende Bibliothek von Bewegungsdaten von ihrer Keyframe-Repräsentation vollständig in ein neuronales Netz übersetzt, welches dann nicht nur die Bewegungsteuerung übernehmen kann, sondern gleichzeitig die Einkopplung sensorischer Signale zulässt.

6.1 Verwendung interpolierter Keyframes

Einen kleinen Ausschnitt aus einer realen Bibliothek von Bewegungsabläufen zeigt Abbildung 6.3. Die numerierten Quadrate stellen Keyframes dar und die Pfeile dazwischen sind Transitionen von einem zum nächsten Key-

frame. Unterschiedlich eingefärbte Rechtecke signalisieren Gruppen gleichartiger Bewegungsabläufe, wie beispielsweise vom Boden aufzustehen, zur Seite zu springen, oder sich nach vorne fallen zu lassen. In Abbildung 6.2 sind zwei Keyframes und eine Transition zu sehen.

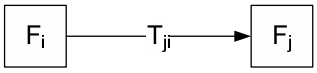

Abbildung 6.2: *Die beiden Keyframes F_i und F_j werden verwendet, um aufeinander folgende Roboterposen zu speichern. Die Interpolation zwischen den Keyframes findet während der Dauer $d_{T_{ji}}$ der Transition T_{ji} statt.*

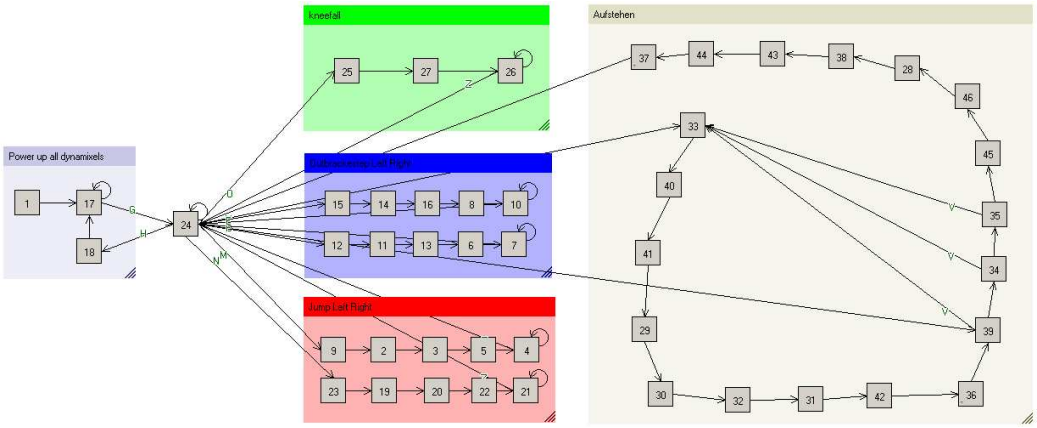

Abbildung 6.3: *Typisches Netzwerk von Keyframes, das unterschiedliche Bewegungsabläufe beinhaltet. Eingefärbte Gruppierungen zeigen Bewegungen der gleichen Art an, zum Beispiel sind Sprünge zur linken und rechten Seite des Roboters innerhalb des roten Rechtecks zu finden.*

Angenommen, Keyframe F_i definiert die Zielwinkel für alle Gelenke des humanoiden Roboters zu einem gewissen Zeitpunkt t_n. Dann definiert Keyframe F_j die Zielwinkel nach der Transition T_{ji} zum Zeitpunkt $t_{n+1} = t_n + d_{T_{ji}}$. Während der Transition werden alle Zielwinkel linear interpoliert. Dies stellt keine Einschränkung dar, denn die Mechanik des Roboters würde jede kompliziertere Interpolationsfunktion sowieso weitestgehend nivellieren. Nichtlineare Verläufe niederer Frequenz können hingegen einfach angenähert werden, indem zusätzliche Keyframes eingefügt werden.

Selektive Verzweigungen

Von Zeit zu Zeit muss ein Roboter sein Verhalten ändern, zum Beispiel ist es wünschenswert, dass ein humanoider Roboter selbständig wieder aufsteht,

nachdem er zu Boden gefallen ist. Dies kann über selektive Verzweigungen erreicht werden, wie in Abbildung 6.4 gezeigt.

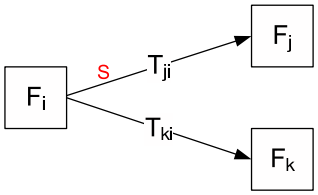

Abbildung 6.4: *Mehrere Transition können denselben Keyframe verlassen. Hier verbindet die unbeschriftete Transition T_{ki} (Standard-Transition) Keyframe F_i mit Keyframe F_k. Die zweite Transition T_{ji} ist optional und mit einem sogenannten Selektor S beschriftet.*

Notwendigerweise kann zu jedem Zeitpunkt stets nur ein Ziel verfolgt werden, daher ist stets maximal ein Selektor aktiv. In jedem Keyframe ist zusätzlich ein boolscher Marker vorgesehn. Falls dieser gesetzt ist, wird der aktive Selektor gelöscht. Dies geschieht stets am Ende in sich geschlossener Bewegungsabläufe, beispielsweise wenn der Roboter nach dem Aufstehen zurück in den Stand gelangt. Anderenfalls würde der Roboter im Stehen erneut mit der Aufstehsequenz beginnen.

6.2 Vom Keyframe zum neuronalen Monoflop

Wie Rumelhart und McClelland in [RM86] eingehend dargestellt haben, lassen sich in einem einfachen Feed-Forward-Netz problemlos mehrere Reiz-/-Reizantwort-Paare speichern, das heißt zu einer bestimmten Eingabe wird die zugehörige Ausgabe generiert. Wenn eine Reihe von ausgegebenen Reizantworten gleichzeitig als Reiz eines anderen Paares fungiert, dann lassen sich Sequenzen generieren, indem man den Ausgang des neuronalen Netzes zum Eingang zurückführt.

Es lassen sich sogar selektive Verzweigungen realisieren, indem einige dedizierte Eingangsneuronen zur Eingabe des aktiven Selektors hinzugenommen werden. Tiňo et al. diskutieren in [THGC98] auf allgemeinere Weise die Implementierung von endlichen Automaten (*Finite State Machines*) unter Verwendung rekurrenter neuronaler Netze.

Eine dritte Überlegung stammt von Afraimovich et al. (siehe [AZR04]) und basiert auf heteroklinen Sequenzen dynamischer Systeme. Im Vergleich zu den beiden erstgenannten Ansätzen lässt Afraimovichs mathematische

Konstruktion prinzipiell auch die Berücksichtigung zeitlicher Randbedingungen zu. Leider eignet sich das in Analogie zur Neurobiologie zeitkontinuierlich gestaltete Modell nicht für die Echtzeitberechnung auf autonomen Robotern.

Im Gegensatz zu den in [RM86], [THGC98] und [AZR04] dargestellten Ansätzen, werden bei der im folgenden vorgestellten Methode gezielt die Transienten einer zeitdiskreten Neurodynamik eingesetzt. Da bei diesem Verfahren jeder Keyframe durch ein aus zwei Neuronen bestehendes Modul ersetzt wird, erhält man schließlich ein sehr dünn verknüpftes Netzwerk, welches sich einfach implementieren lässt, da die Komplexität mit $O(n)$ für n Keyframes relativ niedrig liegt.

Ziel ist es, eine rein neuronale Entsprechung für existierende und auf Keyframes basierende Bibliotheken von Bewegungsdaten zu finden, wobei präzise zeitlich koordinierte Transitionen und selektive Verzweigungen mit abgebildet werden sollen. Die Aktivierung zwischen Keyframes wird über (gegebenenfalls optionale) Transienten weitergereicht, daher liegt es nahe, dieses Verhalten innerhalb eines neuronalen Netzes nachzubilden, wo die Aktivierung sich über die gewichteten Verbindungen von Neuron zu Neuron überträgt.

Mit Hilfe künstlicher Evolution wurden unterschiedliche Neuromodule gefunden, welche die notwendigen dynamischen Eigenschaften aufwiesen. Zunächst wurden alle Module aussortiert, die nicht robust genug waren, das heißt zu sehr von der exakten Einstellung der Gewichte oder von der exakten Einhaltung des eingehenden Signalpegels abhingen. Von den verbleibenden Modulen wurde das gewählt, beim dem sich der zeitliche Verlauf des Ausgangssignals am einfachsten variieren ließ, nämlich über die Veränderung eines einzelnen Gewichts. Hierbei wurde darauf geachtet, dass sich die Form des Ausgangssignals nur in seiner zeitlichen Ausdehnung, nicht aber in seiner generellen Gestalt ändert. Zuletzt wurden die Gewichte auf eine Nachkommastelle gerundet, was keine Auswirkungen auf die Funktionsweise hatte. Das finale Neuromodul ist in Abbildung 6.5 zu sehen. Die Abbildungsgleichung lautet wie folgt:

$$\begin{pmatrix} x_o(t) \\ x_h(t) \end{pmatrix} := \tanh \left[\mathbf{W} \begin{pmatrix} x_o(t-1) \\ x_h(t-1) \\ x_i(t-1) \\ 1 \end{pmatrix} \right], \tag{6.1}$$

wobei t wie üblich einen diskreten Zeitpunkt darstellt, x_i und x_o repräsentieren Eingangs- und Ausgangssignal, x_h das Signal des versteckten Neurons und

$$\mathbf{W} := \begin{pmatrix} 0.1 & 3.7 & -2.6 & -3.9 \\ w_f & 3.4 & 3.9 & 2.8 \end{pmatrix} \tag{6.2}$$

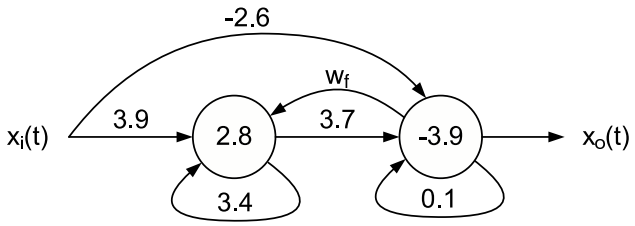

Abbildung 6.5: *Neuronales Monoflop, welches einen Keyframe und eine zu selbigem führende Transition ersetzt. Über das rekurrente Gewicht w_f lässt sich die Zeitkonstante einstellen. Der Detektionsschwellwert des Eingangssignals sowie Pegel und Flankensteilheit des Ausgangssignals bleiben dabei unverändert. Die Zahlen innerhalb der Neuronen bezeichnen die Bias-Werte.*

ist die Gewichtsmatrix, welche den Parameter w_f zur Einstellung der Zeitkonstante sowie die Bias-Werte der beiden Neuronen enthält.

Im Ruhezustand verweilen x_i, x_h und x_o auf einem negativ gesättigten Pegel von nahezu -1. Nachdem das Neuromodul einen positiven Rechteckimpuls beliebiger Dauer empfangen hat, antwortet es seinerseits mit einem positiven Rechteckimpuls am Ausgang. Daher ist es gerechtfertigt, das Modul als neuronales Monoflop zu bezeichnen.

6.3 Funktionelle Analyse im Phasenraum

Sieht man von den kurzen Schaltvorgängen ab, dann ist das Eingangssignal x_i immer konstant fast nahezu -1 oder $+1$. Das dynamische System aus Gleichung 6.1 lässt sich daher wie folgt reduzieren:

$$\mathbf{x}(t) := \tanh\left(\widetilde{\mathbf{W}}\mathbf{x}(t-1) + \mathbf{b}(x_i)\right), \tag{6.3}$$

wobei

$$\mathbf{x} := \begin{pmatrix} x_o \\ x_h \end{pmatrix}, \qquad \widetilde{\mathbf{W}} := \begin{pmatrix} 0.1 & 3.7 \\ -0.5647 & 3.4 \end{pmatrix} \tag{6.4}$$

und

$$\mathbf{b}(x_i) := \begin{cases} (\ -6.5 \quad 6.7\ )^T, & x_i = +1 \\ (\ -1.3 \quad -1.1\ )^T, & x_i = -1 \end{cases} \tag{6.5}$$

in Abhängigkeit des für die Analyse als konstant angenommenen Eingangssignals x_i. Hierbei wurde die Pulsbreite des Monoflops über den Parameter $w_f = -0.5647$ auf 50 Zeitschritte eingestellt. Das Monoflop besitzt für $x_i = -1$ einen einzigen, stabilen Fixpunkt bei $\mathbf{x}^*_- \approx (\ -1 \quad -1\ )^T$ und für $x_i = +1$ einen solchen an der Stelle $\mathbf{x}^*_+ \approx (\ -1 \quad +1\ )^T$.

Ein positiver Rechteckimpuls am Eingang besteht aus einer positiven Flanke, verweilt dann für eine unbekannte Zeit nahe +1 und kehrt schließlich mit einer negativen Flanke zu −1 zurück. Die Flanken sind so steil, dass sie nur ein bis drei Zeitschritte benötigen. In dieser Zeit bewegt sich das System quasi nicht von der Stelle, denn das Vektorfeld

$$\mathbf{V}_{x_i}(\mathbf{x}) := \tanh\left(\widetilde{\mathbf{W}}\mathbf{x} + \mathbf{b}(x_i)\right) - \mathbf{x} \tag{6.6}$$

ist um den jeweiligen Fixpunkt herum betragsmäßig verschwindend klein. Als Bemerkung sei erwähnt, dass das System bei langsamer Variation des Eingangssignals eine nicht triviale Bifurkationssequenz durchläuft, bei der bis zu fünf Fixpunkte koexistieren.

Die Funktionsweise des neuronalen Monoflops lässt sich nun im Phasenraum Stück um Stück darstellen. Wie bereits erwähnt, sind im Ruhezustand alle Signale negativ gesättigt, das Monoflop verweilt also für $x_i = -1$ im Fixpunkt $\mathbf{x}^*_-$. Nach der positiven Flanke des Eingangssignals ergibt sich die in Abbildung 6.6 dargestellte Situation. Anstelle des Fixpunkts $\mathbf{x}^*_-$ ist nun der Fixpunkt $\mathbf{x}^*_+$ vorhanden, welcher innerhalb weniger Zeitschritte erreicht wird. Für beide Fixpunkte gilt $x_o = -1$, das Ausgangssignal bleibt also unverändert negativ gesättigt. Falls das Monoflop sich nicht im Ruhezustand befunden haben sollte, sondern gerade dabei war, einen Ausgangsimpuls zu produzieren, dann wird dieser sofort abgebrochen und der Ausgang auf $x_o = -1$ zurückgesetzt. Wie lange das Eingangssignal positiv bleibt ist völlig unerheblich, solange dies mindestens für fünf Zeitschritte der Fall ist. Wird auf der realen Roboterplattform mit einer Abtastrate von 100Hz gearbeitet (vergleiche Kapitel 11.5), dann muss eine Transition zwischen zwei Keyframes stets mindestens 50ms betragen, damit das neuronale Monoflop verwendet werden kann, was in der Praxis keine Einschränkung darstellt. Die zeitliche Auflösung bleibt über den Parameter w_f stets auf den Zeitschritt genau einstellbar.

Sobald das Eingangssignal mit einer negativen Flanke auf $x_i = -1$ zurückspringt, ändert sich das Vektorfeld wie in Abbildung 6.7 gezeigt. Der Ausgang springt sofort auf $x_o = 1$ und verweilt dort für die mit Parameter w_f eingestellte Zeitdauer (in diesem Fall 50 Zeitschritte).

Robustheit und Einstellung der Zeitkonstanten

Bemerkenswert beim neuronalen Monoflop ist das extreme Verhältnis von schnellem Schaltprozess und langer Verweildauer an quasi derselben Stelle im Phasenraum bei Verwendung von lediglich zwei Neuronen. Die neuronale Realisierung steht im Gegensatz zum üblichen Aufbau eines elektronischen Monoflops, wo als zeitbestimmendes Moment die Entladespannung

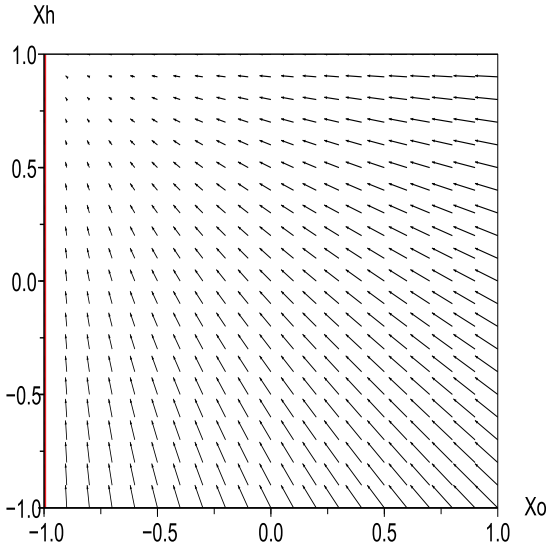

Abbildung 6.6: *Ein positives Eingangssignal $x_i = 1$ aktiviert das versteckte Neuron, das Signal des Ausgangsneurons bleibt jedoch bei -1. Falls das Modul gerade einen Ausgangsimpuls produziert, so wird dieser verkürzt und unmittelbar auf $x_o = -1$ zurückgesetzt. Das Vektorfeld wurde zur Darstellung automatisch so herabskaliert, dass sich eine optimale Ablesbarkeit ergibt. Die Vektorlängen sind also nur relativ zu interpretieren.*

eines Kondensators verwendet wird, welche sich über einen größeren Spannungsbereich erstreckt. Das Ausgangssignal des entsprechenden Integrators wird dann durch den Vergleich mit einer Referenzspannung über einen Komparator mit Schmitt-Trigger gewonnen. Dessen Wirkungsweise wurde bereits in Kapitel 4.2 besprochen.

Das neuronale Monoflop arbeitet hingegen ohne Integrator und Schmitt-Trigger, also auch ohne Hysterese. Die große Zeitkonstante wird hier durch einen sogenannten *Geist* ermöglicht, das heißt einen durch Bifurkation gerade eben verschwundenen Fixpunkt, der aber noch massive Auswirkungen auf den Phasenraum hat, indem er nahegelegene Transienten verlangsamt (siehe [Str94]). Der dazugehörige Bifurkationsparameter ist selbstverständlich w_f; wird der Wert zu groß, dann verwandelt sich der Geist nahe $(\ 1 \quad 1\ )^T$ wieder zurück in einen stabilen Fixpunkt und die Pulsdauer des Monoflops wächst auf unendlich.

Ebenfalls interessant ist der Phasenraum im Bereich $x_o = -1$ und $x_h > 0.5$ (siehe Abbildung 6.7). Die von dort ausgehenden Transienten erreichen den Geist noch. Das Ausgangssignal bekommt daher auch einen positiv ge-

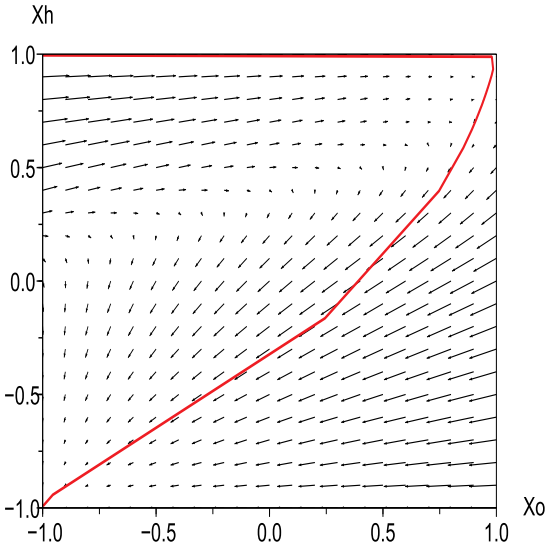

Abbildung 6.7: *Nach einer negativen Flanke des Eingangssignals ($x_i \downarrow -1$) wird der Ausgang unmittelbar auf $x_o = 1$ aktiviert, wo er für lange Zeit verweilt, da das Vektorfeld betragsmäßig fast verschwindet wenn beide Neuronen stark positiviert sind (siehe rechte obere Ecke). Die Rückkehr zur Ruhelage geschieht innerhalb von nur drei Zeitschritten (siehe Kurvenknicke), was eine steile negative Flanke des Ausgangsimpulses sicherstellt (vergleiche auch Abbildung 6.8).*

sättigten Pegel und verweilt für die eingestellte Zeitdauer, selbst wenn das versteckte Neuron durch das Eingangssignal nicht vollständig positiv aufgeladen wurde. Dies erhöht zusätzlich die Robustheit des Moduls gegenüber Eingangssignalen, die extrem kurz sind oder nicht den vollen Signalpegel besitzen.

In Abbildung 6.8 sind die Signalverläufe des neuronalen Monoflops in Abhängigkeit des Parameters w_f zu sehen. Beim Evolvieren und Optimieren des Moduls wurden die Gewichte so gewählt, dass sich der am häufigsten benötigte Bereich von Impulsdauern mit einer Länge von 5 bis 50 Zeitschritten besonders einfach einstellen lässt. Bei einer Abtastfrequenz von 100Hz entspricht dies Transitionen mit einer Dauer von 50ms bis zu einer halben Sekunde. Längere Impulsdauern sind auch realisierbar, benötigen jedoch eine entsprechend höhere numerische Auflösung sowohl des Parameters w_f, als auch der neuronalen Signale selbst.

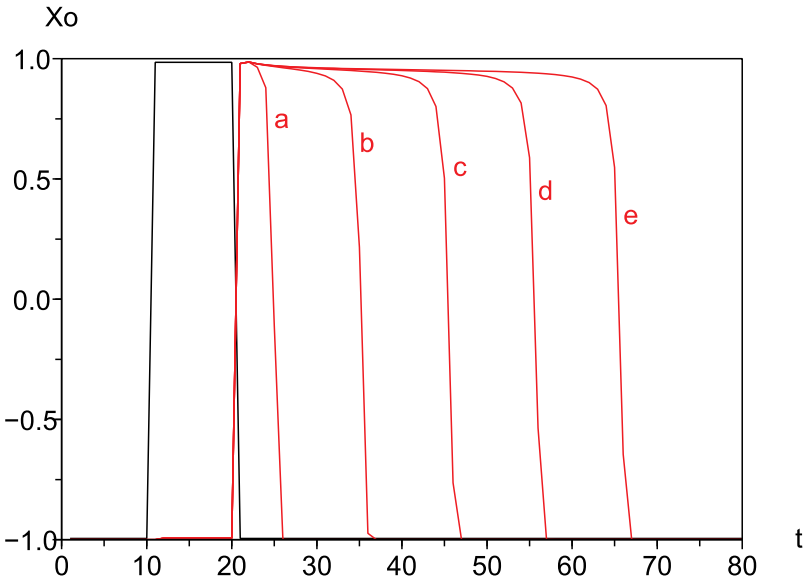

Abbildung 6.8: *Die Pulsbreite des neuronalen Monoflops kann über den Parameter w_f in weiten Grenzen variiert werden. Zu sehen sind die rechteckigen Ausgangsimpulse für folgende Werte: a)* -0.9959*, b)* -0.6087*, c)* -0.5777*, d)* -0.5692 *und e)* -0.5657*.*

6.4 Komposition der Gesamtverschaltung

Um eine auf Keyframes basierende Bibliothek von Bewegungsdaten vollständig in ihre neuronale Entsprechung zu übersetzen, geht man wie folgt vor:

1. Zunächst wird für jedes Keyframe und die zu ihm führende Transition ein zeitlich abgestimmtes neuronales Monoflop eingesetzt. Die Ein- und Ausgänge der aufeinander folgenden Monoflops werden dabei entsprechend des Ursprungsnetzwerks mit Verbindungen vom Gewicht $+1$ versehen.

2. Selektive Verzweigungen benötigen eine Anpassung der Bias-Werte. Für jeden zusätzlichen Eingang zu einem bestimmten Monoflop muss der Bias des versteckten Neurons um $+1$ erhöht werden, um den zusätzlich eingehenden Ruhepegel von -1 auszugleichen.

3. Die Selektoren selbst sind als separate Neuronen vorzusehen, welche im aktiven Zustand den Wert $+1$ haben und im inaktiven Zustand den Wert -1. Gehen von einem Keyframe nun zwei Transitionen aus – eine mit und eine ohne Selektor, dann wird zum neuronalen Monoflop,

das der Transiente mit Selektor entspricht, eine Verbindung vom Selektorneuron mit dem Gewicht +1 gelegt und zum anderen Monoflop (ohne Selektor) eine Verbindung mit dem Gewicht −1. Die Bias-Werte beider versteckten Neuronen müssen jeweils um −1 korrigiert werden. Von den beiden Monoflops wird daher entsprechend der Aktivität des Selektorneurons immer nur eins aktivierbar sein, das andere ist durch die hohe negative Vorspannung blockiert.

4. Vom Ausgangsneuron jedes Monoflops gehen Verbindungen zu denjenigen Motorneuronen, deren Zielwinkel sich beim Durchlaufen des Keyframes verändern sollen. Die Gewichte sind hierbei entsprechend der Zielwinkel zu wählen.

5. Schließlich sind die Bias-Werte der Motorneuronen je nach Anzahl der eingehenden Verbindungen zu erhöhen. Über Selbstkopplungen bei den Motorneuronen erreicht man das in Kapitel 3.1 beschriebene Integrationsverhalten, wodurch die Interpolation während der Zeitdauer der Transitionen respektive der Impulsdauer der Monoflops sichergestellt wird.

Selbstverständlich empfiehlt es sich, das beschriebene Verfahren automatisiert durchzuführen, da eine maschinelle Übersetzung schneller und fehlerfrei erfolgen kann. Außerdem ist es nicht notwendig, irgendwelche Parameter von Hand zu optimieren.

Vorteile gegenüber der Keyframe-Technik

Nach erfolgter Übersetzung einer existierenden Bibliothek in ein entsprechendes neuronales Netz liegt zunächst eine Struktur vor, die das gleiche leistet wie die ursprüngliche Keyframe-Technik. Statt verlinkte Datenstrukturen hat man nun eine große Gewichtsmatrix vor sich, in der alle Bewegungssequenzen kodiert sind. An dieser Stelle sei daran erinnert, dass diese Gewichtsmatrix nur dünn besetzt ist – die Anzahl der von Null verschiedenen Einträge steigt nur linear mit der Anzahl der Keyframes.

Welche Vorteile bietet nun die rein neuronale Steuerung von komplexen Bewegungen? Selbstverständlich lassen sich auf einfache Weise sensorische Signale neuronal einkoppeln. Man kann versuchen, mit Hilfe geschickter Beobachtungen am realen Roboter und etwas Fingerspitzengefühl manuell Einkopplungen vorzunehmen. Beispielsweise liegt es nahe, bei humanoiden Robotern ein Vorbeugen des Oberkörpers zu kompensieren, indem das Hüftgelenk oder die Fußgelenke leicht in die entgegengesetzte Richtung angesteuert werden. Doch dies wäre auch mit einer hybriden Steuerung möglich, bei welcher

zwischen der Keyframe-Ansteuerung und den Motoren eine Schicht von Motorneuronen eingefügt wird, über welche sensorische Einkopplungen erfolgen können.

Der wesentliche Vorteil bei der hier vorgestellten Lösung ist die Möglichkeit, durch Sensoreinkopplung komplexere Mechanismen zur Stabilisierung der Bewegungsverläufe zu bewerkstelligen. Unterschiedliche Mechanismen können dabei im Sinne einer Kaskadenregelung miteinander kombiniert werden. Je nach Ort und Intensität der Einkopplung sind dabei unter anderem folgende Bewegungsmodifikationen realisierbar:

- Direkte Beeinflussung von Gelenkwinkeln
- Verlangsamung oder Beschleunigung von Bewegungssequenzen
- Verkürzung von Bewegungen, Überspringen von Teilbewegungen
- Auswahl von Bewegungsalternativen (kleiner/großer Schritt)
- Veranlassung von bestimmten Bewegungen (Reflexe)
- Unterdrückung ganzer Bewegungssequenzen (Schutzstarre)

Weitere Optionen sind denkbar. Jedoch stellt sich unmittelbar die Frage, wie diese theoretischen Möglichkeiten praktisch ausgeschöpft werden können. Hier bietet sich die künstliche Evolution an, mit der auch schon die Neuromodule selbst gefunden wurden. Dass derart anspruchsvolle Probleme mit evolutionären Verfahren gelöst werden können ist bekannt – entsprechende Publikationen zu kombinatorischen Handlungssequenzen gibt es beispielsweise von Nishimoto und Tani (siehe [NT04]). Nolfi und Parisi beschreiben in [NP99] die Evolution sensomotorischer Koordinationsfähigkeiten und im Rahmen einer laufenden Studienarbeit konnte bereits innerhalb der in Abbildung 6.9 gezeigten Simulationsumgebung (siehe auch [Hei07]) ein humanoider Roboter durch eine neuronale sensomotorische Schleife und einen zusätzlichen homöostatischen Regelmechanismus in stabile Laufbewegungen versetzt werden.

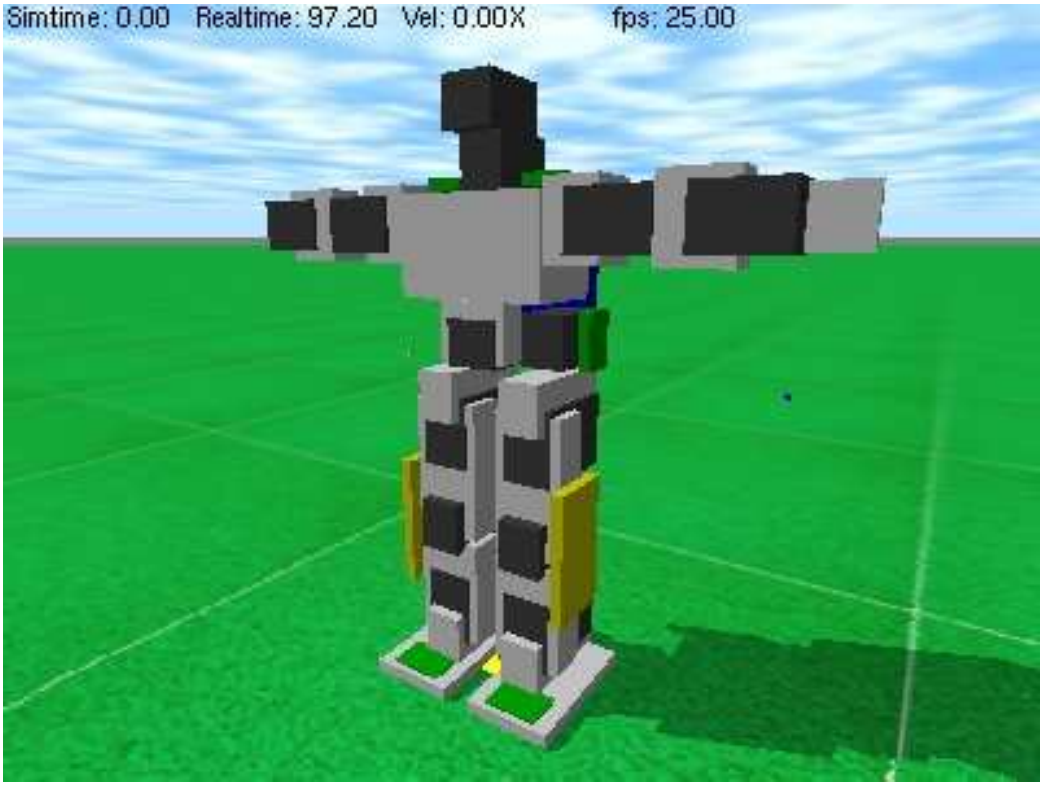

Abbildung 6.9: *Simulationsumgebung, in der Experimente mit humanoiden Robotern, neuronalen Netzen und evolutionären Verfahren gemacht werden können.*

Kapitel 7

Zusammenfassung des theoretischen Teils

Im theoretischen Teil der vorliegenden Arbeit wurden rekurrente neuronale Netze im Hinblick auf ihre Eignung zur Bewegungssteuerung autonomer Roboter untersucht. Hierbei wurde ein im Komplexitätsgrad ansteigender Kanon durchlaufen, beginnend beim Einzelneuron, über verschiedene, aus zwei Neuronen bestehende Module, bis hin zu großen, aus $2n$ Neuronen bestehenden Netzen. Bei den einzelnen Strukturen wurden jeweils die folgenden vier Bereiche behandelt:

1. Mathematische Analyse
2. Numerische Simulation
3. Vergleich mit existierenden Ansätzen
4. Implementierung auf realen Robotern

Es konnte gezeigt werden, dass kleine neuronale Strukturen bereits ausreichend komplexe Neurodynamiken aufweisen, um die adressierten Aufgabenstellungen zu lösen. Dies liegt an der Nichtlinearität der neuronalen Transferfunktion, den rekurrenten Verbindungen sowie an der Verwendung zeitdiskreter Modelle. Während zeitkontinuierliche Modelle erst ab drei Dimensionen chaotisches Verhalten aufweisen, ist dies bei zeitdiskreten Modellen und entsprechender Nichtlinearität schon in einer Dimension möglich. Das erklärt, warum die vorgestellten Module allesamt mit zwei Neuronen auskommen.

Die erste Stufe von bewegungssteuernden Neuromodulen bilden die Oszillatoren – allen voran der SO(2)-Oszillator, aber nicht minder interessant der Magic Circle-Oszillator mit seiner verbesserten Frequenzauflösung und

Parametrisierbarkeit im niederfrequenten Bereich. Die Oszillatoren wurden auf diversen Roboterplattformen erfolgreich getestet und es konnte im Rahmen einer betreuten Diplomarbeit (siehe [Hei07]) gezeigt werden, dass der SO(2)-Oszillator nach Evolution der motorischen Kopplungsgewichte auch einen humanoiden Roboter laufen lassen kann. Es sei bemerkt, dass die dabei entstandenen Laufbewegungen eine ansprechende Ästhetik aufweisen, was sowohl auf die Laufbewegungen selbst zutrifft, als auch auf die natürlich wirkenden Bewegungen aus dem Stand heraus. Ebenso erwähnenswert sind die Resultate mit dem Laufroboter TED, bei dem die Auswirkungen unterschiedlicher Amplituden- und Phasenverhältnisse zwischen den beiden Neuronen eines Moduls systematisch getestet und dokumentiert wurden. Wie sich gezeigt hat, ist über entsprechende Veränderungen der Gewichte Laufen in jede Richtung möglich, inklusive auf der Stelle laufen.

Segmentierte Roboter mit vielen Beinpaaren können mit Hilfe der selbstregulierenden Ringmodule angetrieben werden. Die Herleitung einer homöostatischen Lernregel erfolgte in Anlehnung an biologische Organismen, wo etliche homöostatische Regelprozesse parallel ablaufen, um die Bewegungstüchtigkeit des Organismus zu erhalten. Die Ringmodule sind eine Erweiterung der eben beschriebenen ersten Stufe und weisen neben ihrer eigentlichen Funktionsweise unerwartete Eigenschaften auf. Die Möglichkeit, lokal in einem Körpersegment, ohne Verwendung zusätzlicher Signalpfade, Rückschlüsse auf den Rest des Körpers ziehen zu können, kann als propriozeptive Wahrnehmung betrachtet werden. Die vorgestellte dezentrale homöostatische Bewegungssteuerung ist konzeptionell vollständig neu und in der Literatur ist bis dato kein vergleichbarer Ansatz zu finden.

Die dritte Stufe von Bewegungssteuerung betrifft komplexe Bewegungsabläufe. Damit ist einerseits die gleichzeitige Ansteuerung vieler Freiheitsgrade gemeint, andererseits bezieht sich das Adjektiv „komplex“ auch auf die Motorsignale selbst. Um einen humanoiden Roboter so anzusteuern, dass er sich vom Rücken auf den Bauch dreht, aufsteht, ein paar Schritte läuft und dann einen Ball kickt, bedarf es deutlich mehr als periodische Schwingungen. Dies alles lässt sich mit den vorgestellten monostabilen Neuromodulen realisieren. Es wurde dargestellt, wie sich eine Bibliothek bestehender, manuell gestalteter Bewegungssequenzen mit Hilfe neuronaler Monoflops in ein neuronales Netz übersetzen lässt. Hierbei gewinnt man die Möglichkeit, über sensorische Einkopplungen weitreichende Bewegungsmodifikationen zu realisieren. Diese Vorteile voll auszuschöpfen bedarf sicherlich evolutionärer Verfahren, doch es wurde ausführlich gezeigt, dass die monostabilen Neuromodule die notwendigen Neurodynamiken bereitstellen.

Zusammenfassend lässt sich festhalten, dass mit den in vorliegender Arbeit beschriebenen neurodynamischen Modulen die Bewegungssteuerung au-

tonomer mobiler Roboter unterschiedlichster Morphologien bewerkstelligt werden kann. Die Module lassen sich selbst auf einfachen Plattformen implementieren und benötigen keine hohe Rechengenauigkeit. Über diverse Parameter lassen sich wesentliche Bewegungseigenschaften, wie Laufgeschwindigkeit, Richtung und die Bewegungsauswahl steuern. Aufgrund der reichhaltigen Neurodynamiken eignen sich die Module gut im Zusammenhang mit evolutionären Verfahren. Dabei ist bemerkenswert, dass die Einkopplung eines Bias-Werts entweder ein sich schnell veränderndes Signal sein kann (beispielsweise Filterung von sensorischen Beschleunigungsdaten), oder ein sich langsam verändernder Parameter (zum Beispiel Veränderung der Laufrichtung zu einem sensorischen Reiz hin), oder ein entscheidungsbahnender Signalpegel, der nur zu bestimmten Zeitpunkten wichtig wird (Selektion der gewünschten Bewegungssequenz). Es ist also keine Trennung zwischen reaktiven sensormotorischen Kopplungen, modulierenden Parametern oder symbolischen Werten notwendig.

Teil II

Praxis

Kapitel 8

Übersicht der gebauten Systeme

Um theoretische Erkenntnisse auf ihre Richtigkeit zu überprüfen, ist es notwendig, sie auch in der praktischen Anwendung zu testen. Dies gilt in besonderem Maße für Robotersysteme, die ihr echtes Verhalten erst zeigen, wenn die Interaktion mit der realen Umwelt gegeben ist und die neuronale Steuerung ihre Eingangssignale nicht mehr aus einer stets idealisiert modellierten Simulationsumgebung erhält.

Erst hier zeigt sich, ob das neuronale Netz die notwendige Robustheit besitzt, seine Aufgabe auch noch mit verrauschten und messfehlerbehafteten Signalen zu verrichten. Analoges gilt für die motorische Steuerung, denn Drehmomente, Reibungskoeffizienten, Masseträgheiten, Schwerpunktverlagerungen und dergleichen mehr können nie exakt simuliert werden, da bereits einzelne Parameter nicht-lineare Verläufe aufweisen, ganz zu schweigen von den Wechselwirkungen zwischen mehreren.

Außerdem gibt es Sensorqualitäten, deren Simulation eine umfangreiche Forschungsarbeit für sich allein in Anspruch nehmen würde, wie zum Beispiel die Modellierung der Schallreflexionen in einer Ohrmuschel (siehe hierzu Phonotaxis, Kapitel 9.4).

Ein weiterer Grund für den Bau eines Robotersystems liegt in dessen unmittelbarer Erfahrbarkeit. Dies bezieht sich einerseits auf die Morphologie des Systems: die interaktive Variation von Beinformen an einer Laufmaschine und anschließende Beobachtung der Auswirkungen liefert wichtige Erkenntnisse für den Entwurf der nächsten Generation. Andererseits ist auch die inherente Eigendynamik des Systems von Interesse: wer kann schon mit Sicherheit vorhersagen, wie sich die Verhaltensweise eines autonomen Roboters mit abnehmender Batterieladung allmählich verändert (siehe hierzu die Ergebnisse in Kapitel 10.1 auf Seite 105).

Stammbaum

Aus den genannten Gründen sind im Laufe von fünf Jahren über zehn Robotersysteme und Experimentalaufbauten entstanden, die in den folgenden Kapiteln vorgestellt werden. Eine Übersicht gibt der Stammbaum in Abbildung 8.1. Die Systeme sind chronologisch von oben nach unten angeordnet. Die Pfeile geben an, wo Konzepte weiterverwendet werden konnten, sei es direkt oder in modifizierter Form.

Es lassen sich drei Kategorien nach Mobilitätsgrad voneinander abgrenzen. Im oberen Bereich findet man die stationären Systeme – in der Regel Experimentalaufbauten, die der Untersuchung einer isolierten Fragestellung dienen (Universalgreifer, Künstliches Neuron, Bewegungswahrnehmung und Phonotaxis). Oben und in der Mitte links sind die radgetriebenen Roboter angeordnet (Fahrende Platine und Do:Little) und rechts sowie im unteren Bereich befinden sich schließlich die Laufmaschinen (Lucy, TED, Krabbelroboter, Oktavio und die Humanoide A-Serie).

Die meisten Systeme sind zu komplex, um hier vollständig und detailliert beschrieben werden zu können. Sie werden daher jeweils kurz vorgestellt und anhand eines Blockschaltbildes in ihrer Funktionsweise erläutert. Pro System werden jedoch ein bis zwei besondere Merkmale hervorgehoben und eingehender behandelt.

Obwohl nur die Hälfte der Systeme unmittelbar zu den Laufmaschinen gezählt werden kann und somit in direktem Zusammenhang zum theoretischen Teil der vorliegenden Dissertationsschrift steht, werden im praktischen Teil dennoch alle anderen aufgebauten Systeme mit beschrieben. Einerseits, weil Konzepte dieser Systeme auch bei den Laufmaschinen mit verwendet wurden, andererseits, weil doch an einigen Stellen des theoretischen Teils auf Anwendungen mit diesen Systemen verwiesen wird.

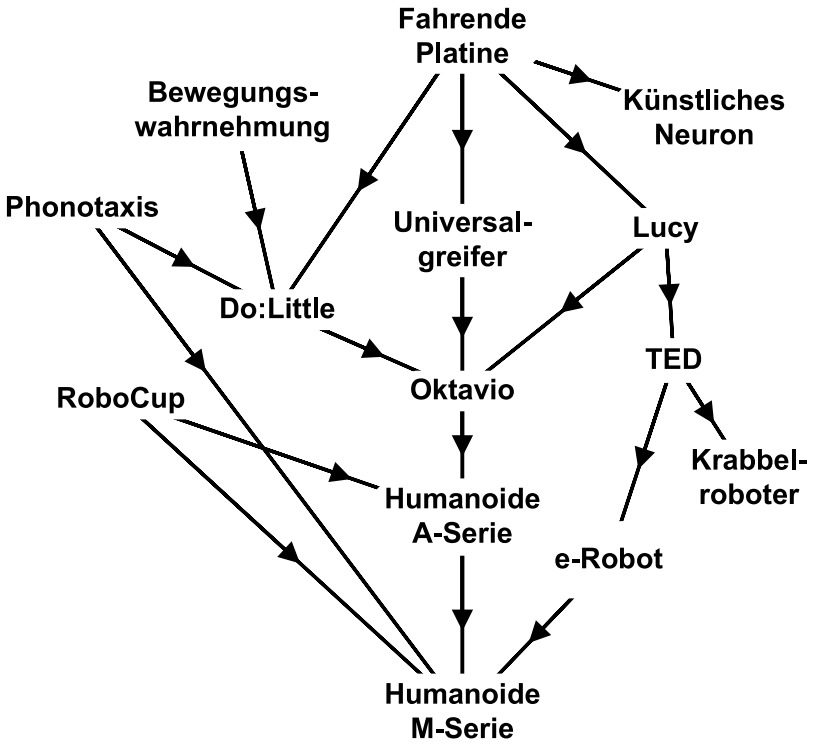

Abbildung 8.1: *Stammbaum der im Laufe von fünf Jahren aufgebauten Experimentalsysteme und Roboterplattformen. Die M-Serie befindet sich noch in der Konzeptionsphase. RoboCup bezeichnet kein System für sich, sondern deutet die Übernahme von Konzepten und Softwaremodulen an.*

Kapitel 9

Stationäre Systeme

In diesem Kapitel werden vier sehr unterschiedliche Systeme zusammengefasst, weil sie zwei Dinge gemeinsam haben: sie sind nicht mobil und sie dienen vorwiegend der Untersuchung einer speziellen Fragestellung. Zwei Systeme nehmen hierbei jedoch eine Sonderstellung ein.

Der von Torsten Siedel entwickelte und 2002 bereits fertig aufgebaute mechanische Universalgreifer wurde um Sensorik und elektronische Steuerung erweitert. Er ist einerseits als in sich geschlossenes System einsatzbereit, kann andererseits aber auch an einen Computer angeschlossen und somit ferngesteuert werden.

Das Künstliche Neuron kommt in keinem der anderen Systeme zum Einsatz, ist hier aber trotzdem mit aufgenommen, weil es zeigt, dass es nicht zwangsweise eines Mikroprozessors bedarf, um die im theoretischen Teil beschriebene nicht-lineare Sättigungsfunktion (Kapitel 2.2) nachzubilden.

9.1 Künstliches Neuron

Das abstrakte Modell eines Neurons besitzt mehrere Eingänge und einen Ausgang. Die Eingangssignale können exitatorisch oder inhibitorisch sein und werden gewichtet zu einer sogenannten Aktivierung aufsummiert. Das Ausgangssignal ist im wesentlichen proportional zu dieser Aktivierung, erreicht aber bei stärkerer Aktivierung zunehmend seinen Sättigungsbereich.

Während im theoretischen Teil durchweg mit diskreten Zeitschritten gearbeitet wurde, welche bei der digitalen Simulation eines Neurons unumgänglich sind, arbeiten analoge elektronische Schaltungen in kontinuierlicher Echtzeit. Neben der naheliegenden Kodierung von Signalstärken durch Spannungspegel, gibt es noch die am biologischen Vorbild angelehnte Möglichkeit der Frequenzmodulation.

Schaltungsbeschreibung

Die hier vorgestellte Schaltung (Abbildung 9.1) arbeitet nach diesem Prinzip. Sie benötigt nur wenige diskrete Bauelemente und arbeitet zuverlässig im Betriebsspannungsbereich von 6–12V.

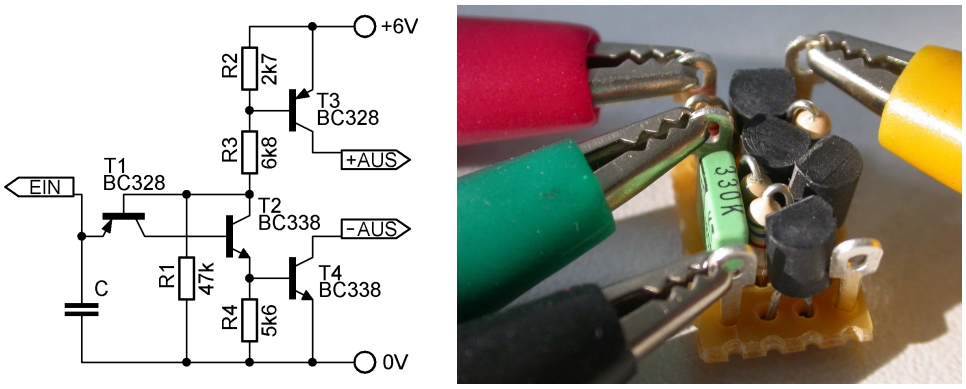

Abbildung 9.1: *(links) Schaltung eines elektronischen Neurons mit diskreten Bauelementen. (rechts) Fertig aufgebautes Neuron mit* $C = 330pF$.

Die eingehenden Impulse laden bzw. entladen den Kondensator C je nach Ausprägung des Signals und Größe des Eingangsgewichts mehr oder weniger. Für exitatorische Verbindungen zum Neuron wird der positive Ausgang des Vorgängerneurons verwendet, für inhibitorische der negative. Die Signalgewichtung geschieht durch einen Widerstand gewünschter Größe (z.B. 100kΩ), der mit einer Diode in Serie geschaltet wird. Die Diode verhindert, dass sich die Ladung des Kondensators C über die Eingangswiderstände mit den Kondensatorladungen benachbarter Neuronen ausgleicht.

Überschreitet die Kondensatorspannung den durch $R1$–$R3$ eingestellten Arbeitspunkt an der Basis von $T1$ um die Basis-Emitter-Spannung von $T1$, dann beginnt $T1$ zu leiten und somit auch $T2$–$T4$. Die beiden Ausgänge werden an die Betriebsspannung bzw. 0V durchgeschaltet und der Kondensator C entlädt sich schlagartig über $T1$, $T2$, $T4$ und $R4$, wonach das Neuron wieder in seinen Ruhezustand verfällt. Die Ruhestromaufnahme wird durch $R1$–$R3$ bestimmt und beträgt bei einer Betriebsspannung von 6V etwas mehr als 100μA. Abbildung 9.2 auf der nächsten Seite zeigt das Ein- und Ausgangssignal sowie den Spannungsverlauf am Kondensator C für den kompletten Vorgang bei Verwendung eines Eingangswiderstands von 47kΩ.

Übertragungscharakteristik

Wie man sieht, benötigt das Neuron fünf Eingangsimpulse bis es selbst aktiv wird. Das Verhältnis von Eingangs- zu Ausgangsimpulsen ist jedoch nicht nur

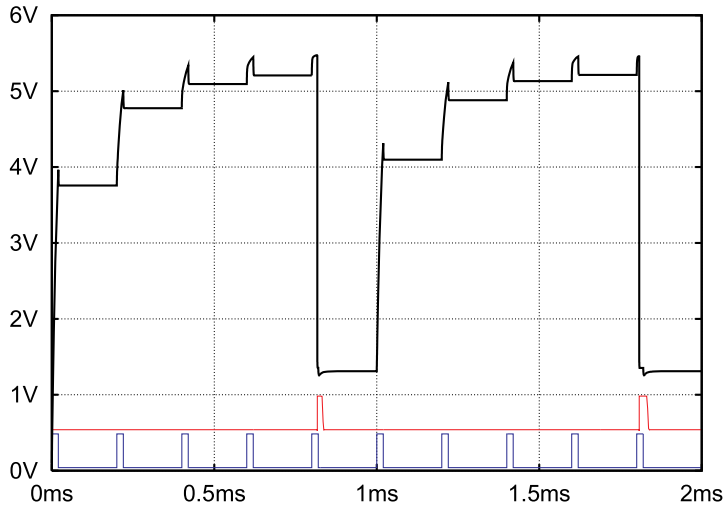

Abbildung 9.2: *Signalverläufe am elektronischen Neuron. Mit jedem Eingangsimpuls (blau) erhöht sich die Spannung am Kondensator C (schwarz) bis sich das Aktionspotenzial des Neurons schlagartig in einen Ausgangsimpuls (rot) entlädt. Die Ein- und Ausgangsimpulse wechseln stets zwischen 0V und 6V, sind jedoch zur besseren Ablesbarkeit gestaucht abgebildet.*

vom Eingangswiderstand abhängig, sondern auch von der Betriebsspannung. Eine Übersicht dieser Abhängigkeit gibt Tabelle 9.1.

	6V	9V	12V
10kΩ	2:1	1:1	1:1
47kΩ	5:1	3:1	2:1
100kΩ	10:1	6:1	5:1

Tabelle 9.1: *Verhältnis von Eingangs- zu Ausgangsimpulsen in Abhängigkeit der Betriebsspannung und des Eingangswiderstands.*

Geht man davon aus, dass ein Robotersystem mit solchen Neuronen und einer Batteriespannung von 12V betrieben würde, dann würde die Empfindlichkeit der Neuronen ungefähr proportional mit der Batteriespannung absinken und die sensorischen Eingangsneuronen würden bei erschöpftem Batteriezustand stärkere Sensorreize benötigen, um erregt zu werden.

Modifikationsmöglichkeiten

Durch die geeignete Wahl des Kondensators C kann der Frequenzbereich des Neurons in weiten Grenzen variiert werden. Die Schaltung wurde erfolgreich mit Kapazitäten von 220pF bis 2.2μF getestet.

Die Schaltung ist zwar schon minimal, lässt sich aber durch die Verwendung von Spezialbauteilen noch weiter vereinfachen. Die beiden Transistoren $T1$ und $T2$ sind als Unijunktionstransistor verschaltet. Setzt man direkt einen solchen anstelle von $T2$ ein, kann $T1$ entfallen.

Desweiteren kann entweder die Kombination $R4/T4$ oder$R2/T3$ entfallen, falls das Neuron ausschließlich exitatorische respektive inhibitorische Verbindungen zu nachfolgenden Neuronen besitzt.

Zum Vergleich: In [GCA] wird ein Chipdesign beschrieben, das 1024 Neuronen mit jeweils 128 Synapsen integriert. Die Neuronen sind von der Funktionsweise her mit dem hier vorgestellten vergleichbar. Der Chip wurde in 0.5μm-CMOS-Technologie gefertigt und verwendet pro Neuron 14 Transistoren und eine Kapazität von nur 88fF (femto $\widehat{=}$ 10^{-15})!

9.2 Universalgreifer

Der Universalgreifer von Torsten Siedel war nach der Fahrenden Platine (siehe Kapitel 10.1) das erste komplexere System, für das es eine Steuerung zu entwerfen galt. Da der Universalgreifer mit einem speziellen Datenhandschuh steuerbar ist, bezieht sich die Komplexität an dieser Stelle auf die große Anzahl von Freiheitsgraden und Sensorsignalen, die es zeitgleich zu verarbeiten galt. Einen ersten Eindruck vom Gesamtsystem erhält man beim Anblick von Abbildung 9.3.

Ziel war es, die vorhandene Mechanik sinnvoll um Sensorik und eine elektronische Steuerung zu erweitern, so dass das System entweder als alleinstehende Funktionseinheit betrieben werden kann, gleichzeitig aber auch die Möglichkeit offen steht, computergestützte Experimente damit durchzuführen.

Nach den positiven Erfahrungen mit der Fahrenden Platine lag es nahe, die wesentlichen Konzepte von dort zu übernehmen. Dies hieß einerseits die Implementierung eines frei konfigurierbaren neuronalen Netzes, in welchem alle Sensordaten münden und das alle Motorneuronen zur Ansteuerung der Freiheitsgrade enthält.

Andererseits bedeutete es aber auch die umfangreiche Nachrüstung des Systems mit zusätzlicher Sensorik – bis dahin war nur ein Drucksensor pro Fingerspitze eingebaut. Hierzu musste viel demontiert werden, denn jeder der

Abbildung 9.3: *(links) Der Universalgreifer ist der menschlichen Hand nachempfunden und besitzt zwölf Freiheitsgrade. (rechts) Die Steuerelektronik ermöglicht nahezu beliebige Kopplungen zwischen den einzelnen Sensor- und Motorsignalen.*

zwölf Servos musste ausgebaut, geöffnet und modifziert werden. Letztendlich wurde jeder Servo mit einem Shuntwiderstand sowie einer Anzapfung des Potentiometers versehen, welches den Istwinkel misst.

Zusammen mit dem interaktiven Datenhandschuh, der neben den Messwertaufnehmern auch mit Servos ausgestattet ist, die es ermöglichen Druck auf die Fingerspitzen des Datenhandschuhträgers auszuüben, ergab sich die folgende Menge von insgesamt 44 Sensor- und 17 Motorsignalen:

- 12 Motorsignale für den Greifer: zwei pro Finger, drei beim Daumen, eins zum Spreizen der Finger
- 32 Sensorsignale vom Greifer: zwei pro Motor (Shunt und Istwinkel), eins pro Fingerspitze (Drucksensor), drei vom Handteller (Drucksensoren in 3-Punkt-Anordnung)
- 5 Motorsignale für den Datenhandschuh: eins pro Fingerspitze (Kraftrückkopplung)
- 12 Sensorsignale vom Datenhandschuh: Dehnungsmessstreifen entsprechend den Freiheitsgraden des Greifers

Designkriterien der elektronischen Steuerung

Das Gesamtsystem sollte schnell und reaktiv arbeiten können, das heißt die langsamste mechanische Komponente allein sollte latenzbestimmend sein. Dies sind die Servos, für deren pulsbreitenmodulierte Steuersignale eine Frequenz von $f = 50\text{Hz}$ vorgeschrieben ist. Daher wurde für die gesamte Signal-

verarbeitung inklusive des neuronalen Netzes eine Zeitscheibe von $T = 20\text{ms}$ pro Berechnungsdurchlauf festgelegt.

Desweiteren sollte das System vom Benutzer unabhängig sein. Aus diesem Grund wurde für die Sensorsignale eine Vorverarbeitungsstufe eingebaut, die eine permanente Autokalibrierung vornimmt. Wie sich später im Test herausstellte ist diese Funktionalität vor allem wichtig, wenn Kinder den Datenhandschuh benutzen, denn kleine Hände können den Wertebereich der Sensoren nicht vollständig ausschöpfen. Durch die Autokalibrierung haben aber auch Kinder den vollen Aktionsradius mit dem Universalgreifer.

Um eine einfache Bedienbarkeit zu gewährleisten, wurde die vollständige Signalverarbeitung im Steuergerät selbst untergebracht. Nach Einschalten des Geräts kann direkt mit dem Universalgreifer agiert werden, da beim Systemstart ein sinnvolles neuronales Netz geladen wird. Das eingebaute Display zeigt alle Sensordaten in Echtzeit grafisch und numerisch an, was eine schnelle Fehlerdiagnose ermöglicht, beispielsweise wenn ein Kabel nicht richtig eingesteckt ist, oder ein Servo klemmt.

Über die Schnittstelle zum PC erweitert sich der Funktionsbereich des Systems schließlich zu einer Experimentalplattform, die es unter anderem ermöglicht mit der gesamten Hardware eine künstliche Evolution durchzuführen (siehe weiter unten).

Funktionsmodule

Wie die Steuerung des Universalgreifers realisiert wurde, zeigt das Blockschaltbild in Abbildung 9.4. Klammert man die Stromversorgung aus, so existieren im wesentlichen noch drei voneinander getrennte Funktionsblöcke.

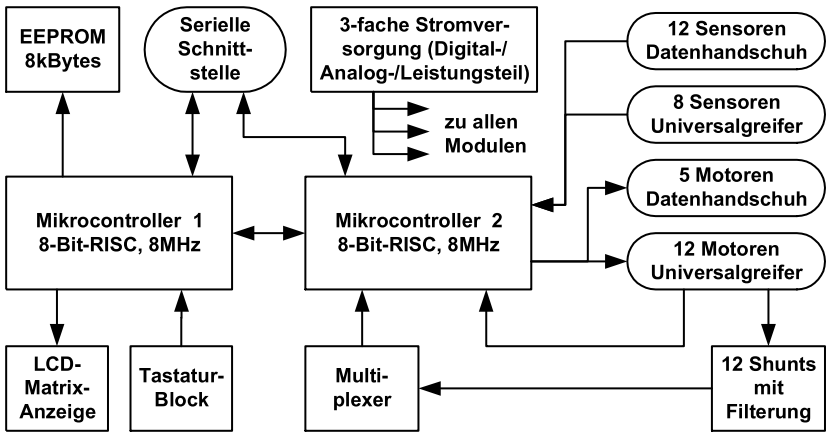

Abbildung 9.4: *Blockschaltbild des Universalgreifers.*

Der erste Block gruppiert sich um den links abgebildeten 8-Bit-RISC-Mikrocontroller 1, der mit 8MHz getaktet wird. Dieser wickelt alle Dienste auf Benutzerebene ab. Er fragt die Tastaturmatrix ab und entprellt die Eingaben, gibt Zeichen und Grafiken auf der LCD-Punktmatrixanzeige aus, kann Einstellungen oder Sensordaten in einem EEPROM abspeichern und lässt sich über eine serielle Schnittstelle mit einem PC verbinden. Über eine Hochgeschwindigkeitsschnittstelle ist der erste Block mit dem zweiten verbunden.

Ein zweiter Mikrocontroller des gleichen Typs bildet den Mittelpunkt des zweiten Funktionsblocks. Hier werden die Sensorsignale erfasst und automatisch kalibriert, das neuronale Netz berechnet und die Servomotoren angesteuert. Über eine zweite serielle Schnittstelle hat man die Möglichkeit, unabhängig vom ersten Mikrocontroller diagnostische Daten mit einem PC auszutauschen, beispielsweise um die Aktivität einzelner Neuronen zu überwachen, oder unter Umgehung des neuronalen Netzes die Motoren direkt anzusteuern.

Den dritten Block bilden die zwölf Shuntwiderstände mit nachfolgender Filterung, Verstärkung und Selektion über einen Multiplexer. Von Aufbau und Funktionstest der umfangreichen Elektronik verschafft Abbildung 9.5 einen Eindruck.

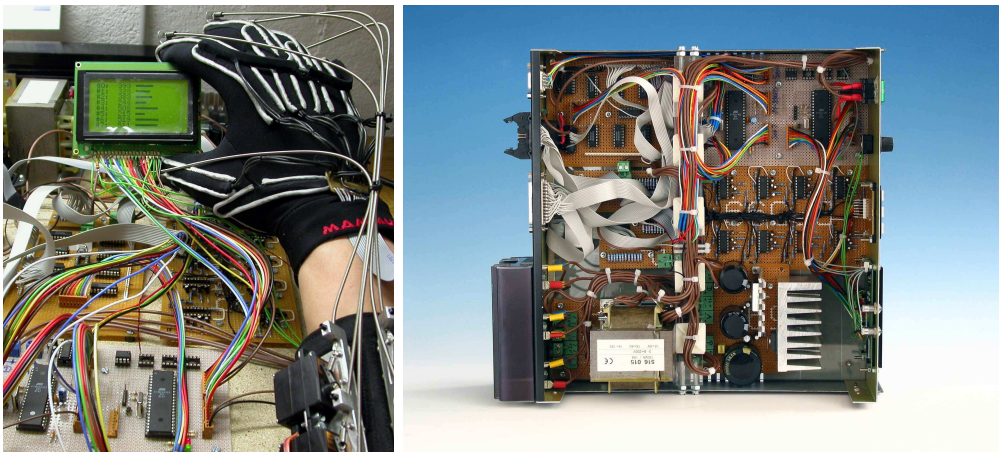

Abbildung 9.5: *(links) Funktionstest der Elektronik vor dem Einbau. Man sieht den Datenhandschuh inklusive der Servomotoren für die Kraftrückkopplung sowie die LCD-Anzeige, auf der die gerade erfassten Sensorsignale des Datenhandschuhs grafisch und numerisch dargestellt werden. (rechts) Blick von oben ins Gehäuseinnere. Das untere Drittel wird von der Stromversorgung eingenommen, links oben befinden sich das Multiplexer-Modul, rechts oben die beiden Mikrocontroller und rechts in der Mitte der Filterblock.*

Bedeutung der propriozeptiven Sensorik

Die Details der elektronischen Auswertung und inhaltlichen Bedeutung der Shuntsignale sind bei der Fahrenden Platine in Kapitel 10.1 auf Seite 103 ausführlich beschrieben. Beim Universalgreifer wurde die Eckfrequenz des Tiefpassfilters wegen der benötigten zeitlichen Auflösung auf $f_0 = 19\text{Hz}$ angehoben.

Durch die zusätzliche Rückführung der aktuellen Winkelpositionen ins neuronale Netz gibt es jedoch gegenüber der Fahrenden Platine beim Universalgreifer erweiterte Möglichkeiten, die Sensorsignale semantisch zu interpretieren. Man betrachte hierzu Abbildung 9.6.

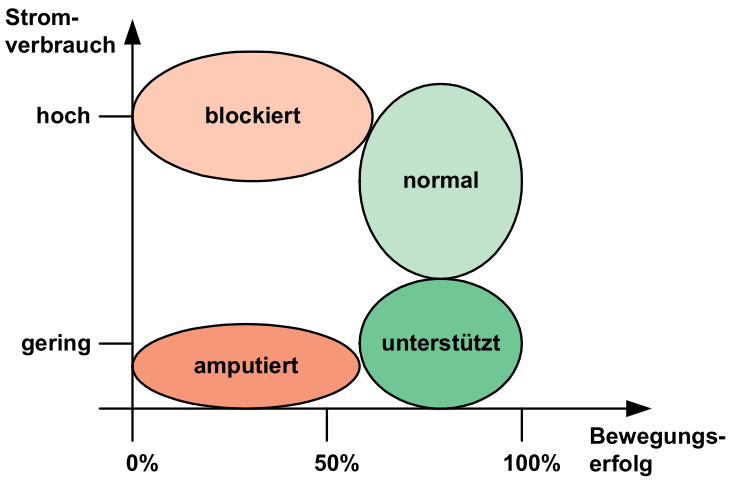

Abbildung 9.6: *Semantische Felder bei Kombination der propriozeptiven Sensorsignale zu einem Freiheitsgrad.*

Im Normalbetrieb erreicht der Servomotor die angestrebte Winkelposition, was sich in einem entsprechenden Stromverbrauch niederschlägt. Wird die Bewegung extern unterstützt, verringert sich der Stromverbrauch deutlich. Falls die Sollposition nicht erreicht wird, kann es dafür zwei Gründe geben: entweder ist der Motor durch einen externen Einfluss blockiert, dann zeigt sich ein hoher Stromverbrauch, oder der Motor ist gar nicht mehr vorhanden, beziehungsweise durchgebrannt, was sich an einem auffällig geringen Stromverbrauch ablesen ließe.

Bemerkung zur sensomotorischen Schleife

In einem Experiment wurde untersucht, was der energetisch günstigste Bewegungsverlauf ist, wenn ein Finger sich aus dem eingerollten Zustand heraus ausstreckt. Mit Hilfe einer künstlichen Evolution (wie in Kapitel 1 auf Seite 6 beschrieben) wurden über hundert Generationen hinweg neuronale Netze auf dem Universalgreifer auf minimalen Energieverbrauch hin evolviert. Dabei wurde vorgeschrieben, dass sich ein Finger innerhalb von fünf Sekunden vom eingerollten Zustand in den ausgestreckten Zustand bewegen muss. Das Kriterium galt als erfüllt, wenn der Finger zum Ende der Zeit im ausgestreckten Zustand angekommen war. Die definierte Fitnessfunktion war dabei umso höher, je geringer das Integral der Stromaufnahme während der fünf Sekunden war.

Das Ergebnis war verblüffend, aber im Nachhinein leicht nachvollziehbar. Man würde eigentlich erwarten, dass die Bewegung langsam beginnt, sich beschleunigt und dann allmählich wieder zum Stillstand kommt, so dass keine grossen Beschleunigungsänderungen notwendig sind. Stattdessen verharrte der Finger ungefähr bis zur Halfte der Zeit im eingerollten Zustand und schnappte dann schnell in den ausgestreckten Zustand, wo er bis zum Ablauf der Zeit blieb.

Wie ist das zu erklären? Ganz einfach: wenn der Servo eine zeitlich längere Bewegung ausführt, verbraucht er während der gesamten Zeit recht viel Strom, weil der Servo bereits einen in sich geschlossenen Regelkreis besitzt, der ständig mit maximaler Energie die Sollposition nachregelt. Es ist also energetisch unsinnig, Servomotoren zur Durchführung langsamer Bewegungen einzusetzen, weil sie nur im absoluten Ruhezustand wirklich wenig Strom verbrauchen. Die künstliche Evolution hatte gar keine Chance, wirklich einen harmonischen Bewegungsverlauf zu evolvieren, weil sie nicht im Regelkreis des Servos lag. Daher war die optimale Lösung, die Regeldauer des Servos so kurz wie möglich zu machen, was die künstliche Evolution auch herausbekam.

So enttäuschend das Resultat vielleicht erscheinen mag, so wichtig war es doch die geschilderte Erkenntnis sehr frühzeitig zu haben. Beim Laufroboter Oktavio (vergleiche hierzu Kapitel 11.4 auf Seite 120), der Energieeffizienz als ein wesentliches Designziel hatte, wurde dies angemessen berücksichtigt, so dass dort alle geschlossenen sensomotorischen Regelkreise stets über das neuronale Netz führen und beliebig modifiziert werden können – sei es von Hand, oder per künstlicher Evolution.

9.3 Bewegungswahrnehmung

Das Experimentalsystem zur Bewegungswahrnehmung wurde gebaut, um eine neuronale Retinastruktur zu evolvieren. Die entsprechenden Resultate sind in der Diplomarbeit von Andreas Wollstein (siehe [Wol07]) zu finden.

Die minimalistisch gehaltene Hardware ist in Abbildung 9.7 zu sehen und recht schnell beschrieben. Auf einem Servo ist eine Schwarz-Weiß-Miniaturkamera montiert. Sie ist waagrecht ausgerichtet und kann vom Servo horizontal in einem Bereich von 180° geschwenkt werden. Kamera und Servo sind an einen Mikrocontroller mit DSP-Funktionalität angeschlossen, auf dem das neuronale Netz berechnet wird.

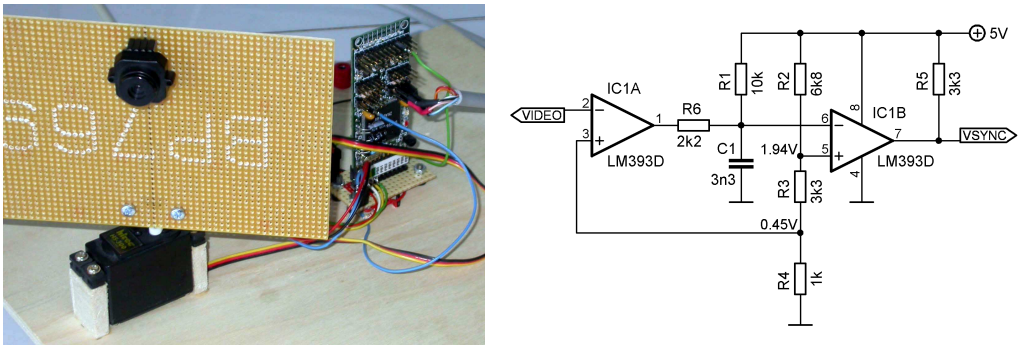

Abbildung 9.7: *(links) Die auf dem Servo montierte Kamera kann horizontal geschwenkt werden. Aus dem Bildsignal entnimmt ein Mikrocontroller mit DSP-Funktionalität einzelne Bildpunkte. (rechts) Schaltung zur Extraktion des vertikalen Synchronsignals.*

Das besondere an diesem System ist die Extrahierung der benötigten Bildinformation. Normalerweise werden bei der Verarbeitung von Bildsignalen immer vollständige Bilder digitalisiert und die benötigten Informationen aus dem Bildspeicher gelesen. Dies erfordert einen nicht trivialen Schaltungsaufwand oder den Kauf eines entsprechenden Standardsystems. Hier wird jedoch ein einfacherer Weg beschritten, da aus dem Bild letztendlich nur wenige Punkte benötigt werden.

Schaltungsbeschreibung

Die Kamera liefert ein sogenanntes Composite-Video-Signal, welches neben der Bildinformation auch sämtliche Synchronsignale für den horizontalen und vertikalen Strahlrücklauf sowie für die Erkennung der Halbbilder beinhaltet. Abbildung 9.8 zeigt einen Ausschnitt des Signals nach der in Deutschland

gültigen Fernsehnorm PAL zum Zeitpunkt des vertikalen Strahlrücklaufs vor Start des ersten Halbbildes.

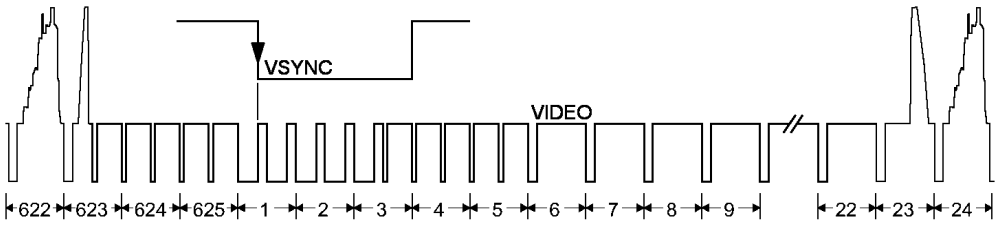

Abbildung 9.8: *Ausschnitt eines PAL-Signals zu Beginn des ersten Halbbildes. Während der Vertikalsynchronisation sind die Synchronimpulse invertiert.*

Wie man sieht, sind die Synchronimpulse in den Bildzeilen Eins und Zwei sowie in der ersten Hälfte der dritten Zeile invertiert, so dass sich der zeitliche Anteil des Nullpegels erhöht. Dies macht sich die Schaltung aus Abbildung 9.7 zunutze.

Bei jedem Synchronimpuls am Schaltungseingang wird die durch den aus $R2$–$R4$ gebildeten Spannungsteiler eingestellte Referenzspannung von 0.45V unterschritten und der Open-Collector-Ausgang von $IC1A$ wird hochohmig. Der bis dahin über den Spannungsteiler $R1/R6$ auf 0.9V gehaltene Kondensator $C1$ beginnt sich über $R1$ positiv aufzuladen. Nur während der langen Nullpegelphasen bei der Vertikalsynchronisation überschreitet die Kondensatorspannung die zweite erzeugte Referenzspannung von 1.94V, wodurch $IC1B$ seinen Ausgang nach Masse kurzschließt. Das Ausgangssignal der Schaltung ist in Abbildung 9.8 mit dargestellt.

Den Rest erledigt der Mikrocontroller des Experimentalaufbaus. Dieser erhält die extrahierten vertikalen Synchronimpulse der eben beschriebenen Schaltung an einem Triggereingang. Über diesen Eingang wird ein interner Zähler gestartet, welcher den Mikrocontroller nach einer voreingestellten Zeit T_1 unterbricht.

Extraktion der Bildpunkte

Die gesamte Bildinformation ist so im Videosignal enthalten, dass ausgehend vom Zeitpunkt des vertikalen Synchronsignals die einzelnen Bildzeilen von oben nach unten durchlaufen werden. Die Bildzeilen selbst werden von links nach rechts abgetastet. Man betrachte hierzu Abbildung 9.9.

Durch gezielte Wahl des Parameters T_1 lässt sich somit exakt die Position des ersten Bildpunkts bestimmen, welcher aus dem Videosignal zu entnehmen

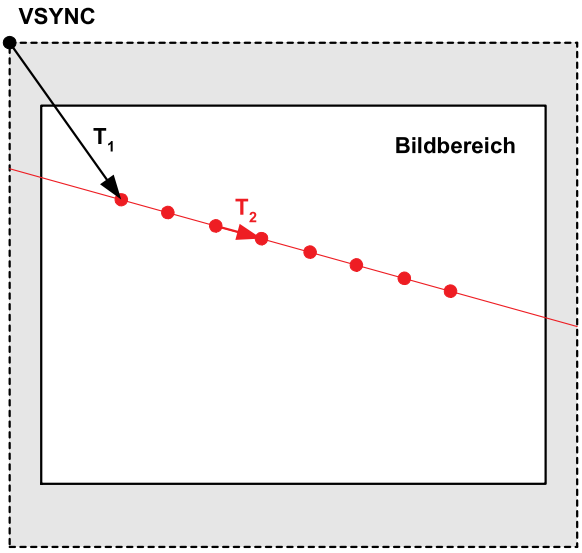

Abbildung 9.9: *Durch die Parameter T_1 und T_2 werden die zu entnehmenden Bildpunkte eindeutig bestimmt.*

ist. Die Tatsache, dass der Bildbereich zwischen verschiedenen Synchronisationsinformationen in das Videosignal integriert ist, stört dabei nicht. Wichtig ist nur, dass die Zeitabstände zwischen bei benachbarten Bildzeilen und die Dauer einer Bildzeile stets konstant sind.

Auf dieselbe Weise bestimmt T_2 den zeitlich Abstand zweier benachbarter Bildpunkte. Es kostet den Mikrocontroller fast keine Rechenzeit die Signalproben zu entnehmen, da er nur zu den vorgegebenen Zeitpunkten $T_1 + nT_2$ das Videosignal über seinen Analog-Digital-Konverter einlesen muss, was zweckmäßigerweise über eine Interruptroutine realisiert wird. Das Hauptprogramm, in welchem das neuronale Netz berechnet wird, bekommt die Helligkeitswerte der Bildpunkte auf diese Weise einfach als mehr oder weniger aktivierte Eingangsneuronen zur Verfügung gestellt.

Über zusätzliche Ausgangsneuronen sowie einfache Abbildungen lässt sich die Anordnung der Punkte im Bildbereich auf vielfältige Weise parametrisieren. Verschiebungen in alle Richtungen lassen sich über T_1 steuern, während über T_2 Streckungen, Stauchungen und Rotationen der Bildpunktlinie relativ zum zeitlich ersten Bildpunkt vorgenommen werden. Mit Hilfe einer Tabelle, die jedem der n Bildpunkte einen Zeitpunkt $T_1 \ldots T_n$ zuordnet, lassen sich beliebige Punktanordnungen umsetzen. Wenn der Roboter seinen Kopf schräg stellen kann, dann ist eine kreisförmige Punktanordnung geeignet, Bewegungen relativ zu diesem Freiheitsgrad zu erkennen.

9.4 Phonotaxis

In Rahmen einer Studienarbeit von Raphael Bauer (siehe [Bau05]) wurden neuronale Netze zur Richtungserkennung von akustischen Signalen evolviert. Die Experimente wurden mit einer recht aufwändigen Versuchsanordnung durchgeführt, deren einzelne Komponenten im nachfolgenden besprochen werden. Im Mittelpunkt steht hierbei die vollständige akustische Übertragungsstrecke, deren Gesamtcharakteristik ausgenutzt werden soll, um die Richtung einer Signalquelle relativ zum Empfänger zu erkennen. Abbildung 9.10 zeigt den Experimentalaufbau des Empfängers.

Auf einem kleinen Holzbrettchen sind zwei künstliche Ohren im Abstand von 15cm befestigt. Der Ohrabstand sowie die Größe der Ohren selbst entspricht durchschnittlichen menschlichen Maßen. Auch sind die Ohren aus elastischem Silikon hergestellt, damit die diffusen Reflexionen in den Ohrmuscheln möglichst mit denen beim Menschen vergleichbar sind.

Am Ende der Gehörgänge sitzen Miniatur-Mikrofonkapseln. Sie sind mit einem Vorverstärker verbunden, welcher zwischen den beiden Ohren montiert ist. So werden die einstreuungsempfindlichen Kabel auf minimaler Länge gehalten.

Die gesamte Einheit kann durch zwei Servos auf- und abgeneigt sowie in einem Bereich von 180° horizontal gedreht werden. Die Servos sind zusammen mit elektronischen Pegelwandlern und den notwendigen Anschlussbuchsen auf einer Grundplatte befestigt, die auch bei ruckartigen Bewegungen der Servos stabil auf dem Untergrund stehen bleibt.

Funktionsblöcke

Um überhaupt sinnvolle Experimente zum Richtungshören durchführen zu können, muss die gesamte Signalverarbeitungskette zwei Kriterien erfüllen:

- Die zeitliche Auflösung bzw. der ausgewertete Frequenzbereich muss ausreichend gewählt werden, damit die Laufzeitunterschiede zwischen linkem und rechtem Ohr ausgewertet werden können.

- Die auswertbare Dynamik – und damit auch der Störspannungsabstand des Systems – muss gross genug sein, damit sowohl laute, nahe Signale, wie auch leise, entferntere Signale zuverlässig entdeckt werden können.

Aus diesem Grund wurde entschieden, die weitere Signalverarbeitung mit dem käuflich fertig erhältlichen, frei programmierbaren, digitalen Signalprozessor Chameleon der Firma Soundart durchzuführen. Das Chameleon ist für

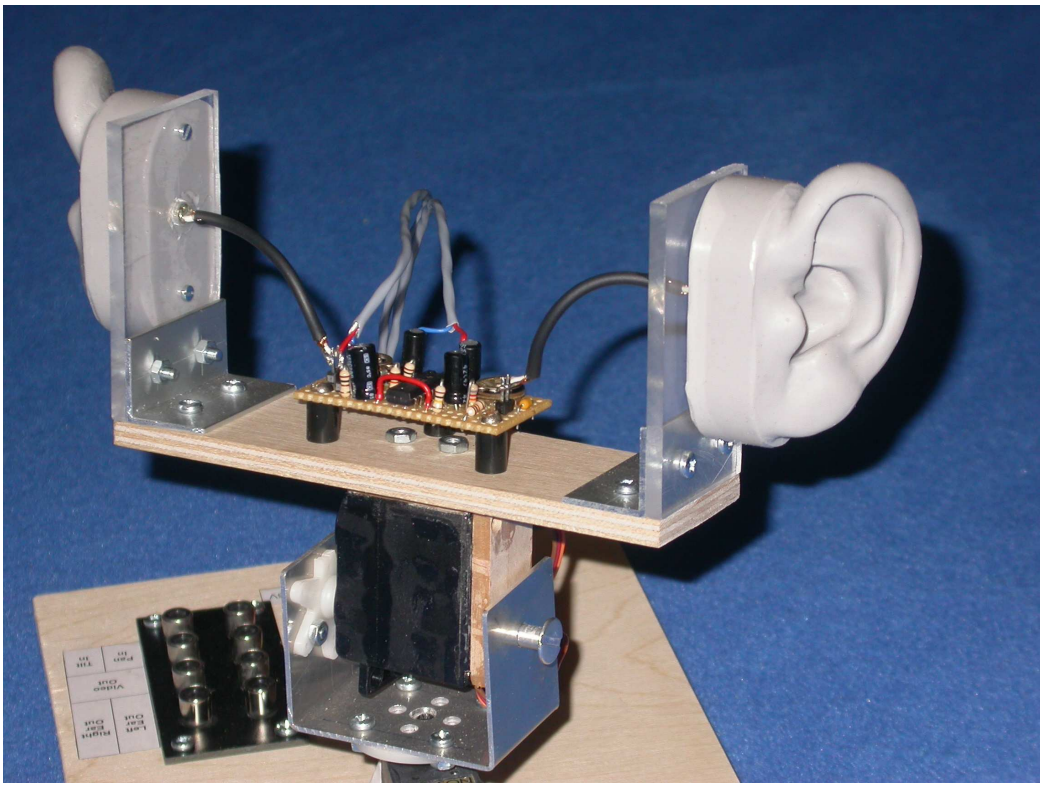

Abbildung 9.10: *Die künstlichen Ohrmuscheln lassen sich horizontal und vertikal schwenken. Am Ende des Gehörgangs sitzt jeweils eine Miniatur-Mikrofonkapsel, die mit dem Vorverstärker verbunden ist.*

den Einsatz als Synthesizer oder Effektgerät im professionellen Tonstudiobereich konzipiert und erfüllt daher mit einer Abtastfrequenz von $f_s = 48\text{kHz}$ und einem 24-Bit-Stereo-Analog-Digital-Wandler mit 256-fachem Oversampling die geforderten Bedingungen. Der abbildbare Frequenzbereich beträgt 20Hz–20kHz (+0/−0.5dB) und der Signal-to-Noise-Ratio liegt über 90dB (gemessen bei 1kHz). Die gesamte Versuchsanordnung ist in Abbildung 9.11 zu sehen.

Da es den digitalen Signalprozessor (DSP) fast keine Rechenkapazität kostet, zwei Rechteckschwingungen derselben Frequenz $f_0 = 50\text{Hz}$ mit unterschiedlicher Pulsbreitenmodulation zu generieren, war die einfachste Lösung das Chameleon auch die beiden Servos steuern zu lassen. Hierzu wurde ein einfacher Pegelwandler mit in den Experimentalaufbau integriert, welcher das gleichspannungslose Ausgangssignal des Chameleon in den Betriebsspannungsbereich der Servos anhob.

Über die MIDI-Schnittstelle (Musical Instrument Digital Interface) wurde das Chameleon an einen PC angeschlossen, auf welchem zwei über TCP/IP-

Protokoll miteinander kommunizierende Programme abliefen.

Der Versuchsablauf sieht wie folgt aus:

1. Der Populationsmanager sendet die Beschreibung eines zu testenden neuronalen Netzes an die Experimentsteuerung.

2. Die Experimentsteuerung übermittelt dieses Netz an den DSP. Dieser beginnt, die Mikrofonsignale einzulesen, das Netz zu berechnen und die Servos anzusteuern. Mit jeder Servoaktualisierung sendet er eine Kopie der Werte, die den Winkelpositionen der Servos entsprechen, an die Experimentalsteuerung.

3. Die Experimentsteuerung gibt in zufälliger Reihenfolge Signale über verschiedene Lautsprecher aus und speichert gleichzeitig die vom DSP empfangenen Winkelpositionen der Servos ab.

4. Die Experimentsteuerung schaltet den DSP ab und sendet einen Fitnesswert an den Populationsmanager, der umso höher ausfällt, je genauer die Winkelpositionen den Richtungen entsprechen, aus denen die Lautsprecher gesendet haben.

Diese vier Schritte wiederholen sich solange, bis der Benutzer mit der Güte der evolvierten Netze zufrieden ist und den Populationsmanger anhält. Die maximale Größe der neuronalen Netze wird vom Datendurchsatz des DSPs bestimmt und liegt bei einer Aktualisierungsfrequenz von $f_r = 48\text{kHz}$ bei 15 vollständig miteinander verbundenen Neuronen.

Der Experimentalaufbau ist bereits dafür vorbereitet, auch die Kamera aus dem Experiment zur Bewegungswahrnehmung mit aufzunehmen. Letztlich sollen die akustische Richtungswahrnehmung und die visuelle Bewegungswahrnehmung auf der Humanoiden M-Serie mit implementiert werden.

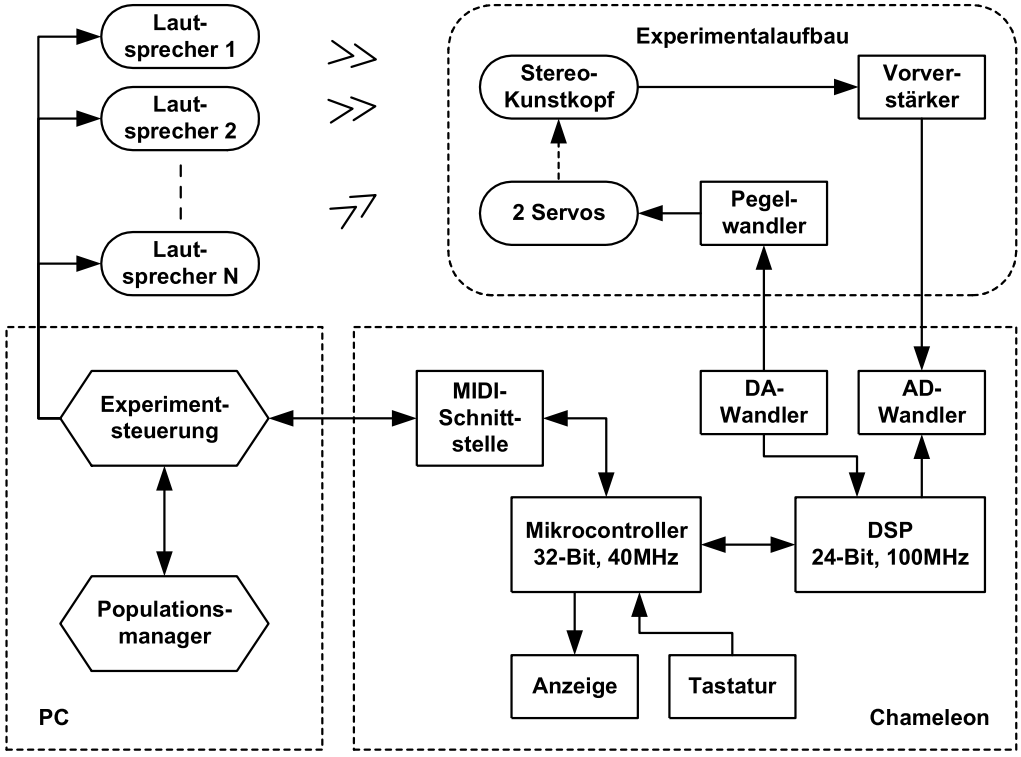

Abbildung 9.11: *Blockschaltbild der Versuchsanordnung. Neben dem Experimentalaufbau aus Abbildung 9.10 kommen noch Lautsprecher, ein PC sowie der frei programmierbare, digitale Signalprozessor Chameleon der Firma Soundart zum Einsatz.*

Kapitel 10

Radgetriebene Roboter

Fahrbare Roboter sind im Vergleich zu Laufmaschinen im mechanischen Aufbau ein Kinderspiel. Der differenzielle Antrieb kann zudem auf einfache Weise angesteuert werden und verleiht dem Roboter die Fähigkeit, sich auf der Stelle zu drehen, was vorteilhaft ist, wenn er sich in einer Sackgasse befindet.

Bei Verwendung eines Ackermann-Antriebs hat man das (zumindest einem Teil der Bevölkerung) bekannte Ein-/Ausparkproblem. Beim omnidirektionalen Antrieb hingegen gewinnt man einen Freiheitsgrad in der Bewegung, benötigt dafür aber auch einen zusätzlichen dritten Motor.

Differenzielle radgetriebene Roboter eignen sich also gut, wenn man rasch eine einfache, mobile Plattform benötigt, an die man allerhand Sensoren montieren und ausprobieren kann. Zudem sind sie wesentlich kostengünstiger herzustellen als Laufmaschinen. Dies ist vorteilhaft, wenn man viele von ihnen herstellen möchte – zum Beispiel um kollektive Verhaltensweisen zu studieren. Aus diesen Gründen sind auch die beiden im folgenden beschriebenen Systeme entstanden.

10.1 Fahrende Platine

Wie aus dem Stammbaum (Abbildung 8.1 auf Seite 84) ersichtlich, ist die Fahrende Platine der Vorgänger aller anderen mobilen Systeme. Beim späteren Vergleich der Blockschaltbilder wird auffallen, dass viele Module beibehalten wurden.

In Abbildung 10.1 ist die erste handschriftliche Skizze der fertig aufgebauten Fahrenden Platine gegenübergestellt und in Abbildung 10.2 auf Seite 103 findet man das Blockschaltbild, in dem alle Komponenten dargestellt sind.

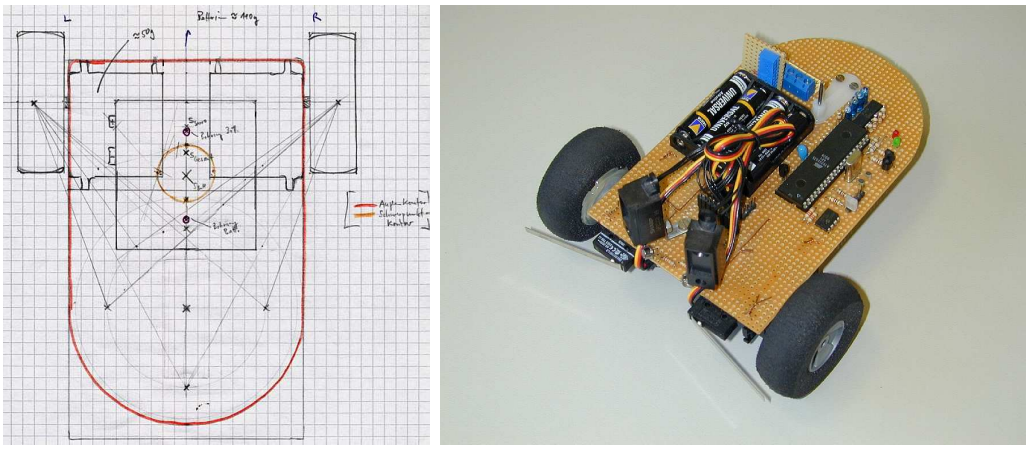

Abbildung 10.1: *(links) Erste handschriftliche Skizze zur Berechnung des optimalen Schwerpunktbereichs. (rechts) Fertig aufgebaute, funktionstüchtige Fahrende Platine.*

Funktionsmodule

Das Herz der Fahrenden Platine ist ein 8-Bit-RISC-Mikrocontroller, der mit 8MHz getaktet wird und zusammen mit der restlichen Elektronik und den beiden Servos aus vier 1.5V-Batterien gespeist wird. Der Mikrocontroller besitzt einen erweiterten Betriebsspannungsbereich, so dass weder 6V Betriebsspannung, noch Spannungseinbrüche durch erhöhten Strombedarf bei extern gebremsten Servos ein Problem darstellen. Über eine separate Spannungsreferenz von 2.5V kann indirekt die Batteriespannung berechnet werden, so dass diese Information als propriozeptiver Sensorwert zur Verfügung steht.

Die drei jeweils im links/rechts-Paar vorhandenen Umgebungssensoren sind schaltungstechnisch einfach:

- Helligkeitssensor: Spannungsteiler mit LDR und Widerstand
- Abstandssensor: Infrarot-Triangulations-Sensor, der über ein digitales Protokoll ausgelesen wird
- Kollisionssensor: Mikroschalter mit Pull-up-Widerstand

Zur Ausgabe von Statusinformationen, bzw. um im laufenden Betrieb interne Zustände mitprotokollieren zu können, gibt es neben zwei Leuchtdioden auch einen Datenfunksender, welcher am Ausgang einer seriellen Schnittstelle angeschlossen ist. Über Erweiterungsbuchsen lassen sich zusätzliche Signale ausgeben oder weitere Sensoren anschliessen.

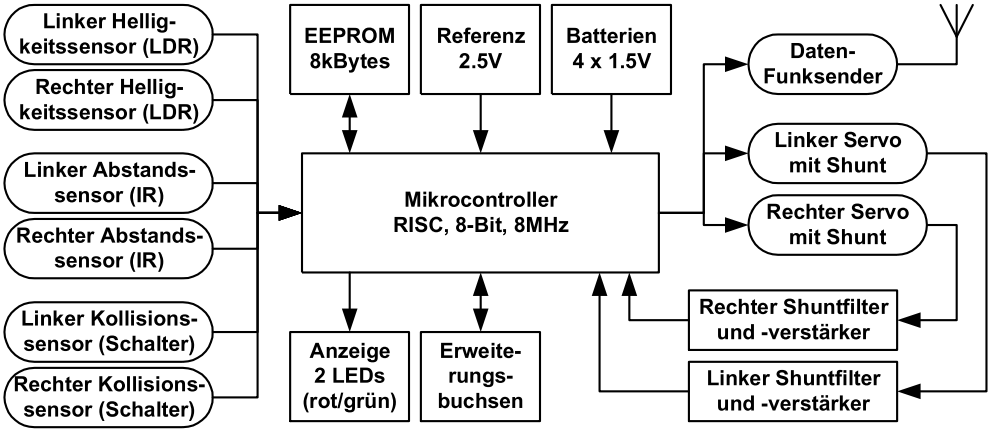

Abbildung 10.2: *Blockschaltbild der Fahrenden Platine.*

Ein separates EEPROM dient der batterieunabhängigen Speicherung von Populations- und Individuen-Daten, wenn eine sogenannte On-board-Evolution durchgeführt wird (wie in Kapitel 1 auf Seite 7 beschrieben).

Antrieb und Motorsensorik

Die Fahrende Platine wird von modifizierten Servos angetrieben, bei denen die Endanschläge entfernt und die Potentiometer durch feste Spannungsteiler ersetzt wurden.

Besonders hervorzuheben ist der masseseitige Einbau von Shunt-Widerständen in die Servo-Stromkreise sowie die Aufbereitung der damit gewonnenen Signalspannungen. Durch diese Maßnahme stehen dem System zwei weitere propriozeptive Sensoren zur Verfügung.

Abbildung 10.3 auf der nächsten Seite zeigt den Schaltplan. Die Servos lassen sich auch mit eingebautem Shunt-Widerstand weiterhin problemlos pulsbreitenmoduliert ansteuern.

Die an $R1$ abfallende Shuntspannung ist proportional zum Servostrom, welcher wiederum streng monoton von der externen Belastung des Servomotors abhängt. Die Shuntspannung wird zunächst mit einem Tiefpassfilter zweiter Ordnung von überlagerten Störsignalen befreit. Das mit $R2$, $R3$, $C1$, $C2$ und $IC1A$ aufgebaute Sallen-Key-Filter hat eine Gleichspannungsverstärkung von $A = 1.0$ und eine Eckfrequenz von $f_0 = 1.9\text{Hz}$, was einer Zeitkonstante von $T = 0.52\text{s}$ entspricht.

Der mit $R4$, $R5$ und $IC1B$ aufgebaute Verstärker hebt das Nutzsignal auf das 57-fache an. Die Verstärkung wurde bewusst nicht im Filter selbst

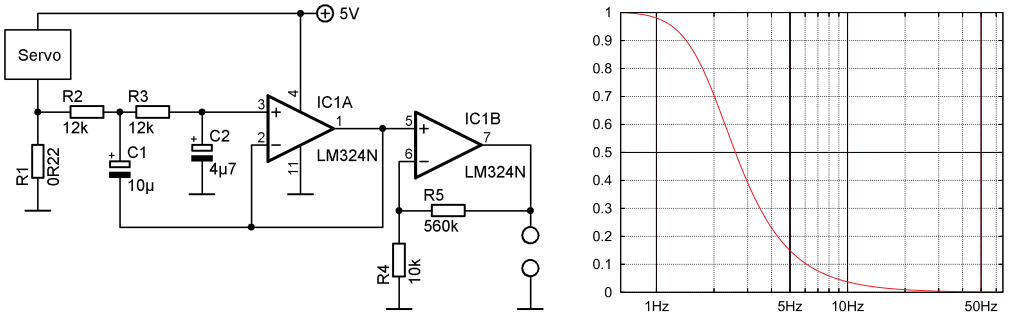

Abbildung 10.3: *(links) Schaltung für die Filterung und Verstärkung des Shuntsignals am Servo. (rechts) Zugehörige Filtercharakteristik, normiert auf die Gleichspannungsverstärkung.*

vorgenommen, um die Schwingneigung des Filters sowie den Einfluss von Bauteiletoleranzen auf die Filtercharakteristik minimal zu halten. Die Filtercharakteristik ist ebenfalls in Abbildung 10.3 mit dargestellt.

Die Wahl des Operationsverstärkers ist in dieser Schaltung besonders kritisch, weil ohne symmetrische Betriebsspannung ein Nutzsignal nahe 0V ausgewertet wird. Der LM324 ist für solche Bedingungen optimal, da er bereits mit einer einfachen Betriebsspannung ab 3V zuverlässig arbeitet und seine niedrigste Ausgangsspannung typischerweise nur 5mV beträgt.

Abbildung 10.4 zeigt das Resultat: Wie man sieht, lassen sich milde und starke Motorbelastungen (1.8–2.2s resp. 1.9–2.9s) deutlich vom Normalbetrieb (3.0–5.0s) unterscheiden.

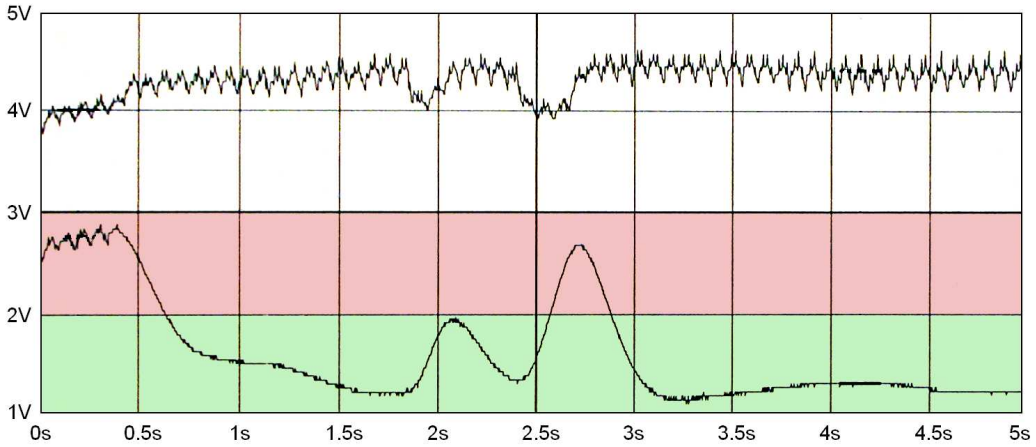

Abbildung 10.4: *Verlauf des aufbereiteten Shuntsignals (untere Kurve) und der Betriebsspannung (obere Kurve) bei unterschiedlichen Motorbelastungen.*

Bemerkung zum Schaltungsdesign

Da die Batterien offensichtlich nicht mehr die volle Ladung besitzen (4.4V bei vier 1.5V-Batterien), sind in den Phasen der Motorbelastung auch stärkere Spannungseinbrüche in der Betriebsspannung zu finden. Diese dient jedoch auch als Referenzspannung für den Analog-Digital-Wandler des Mikrocontrollers, so dass der ausgelesene Skalenwert sich aus dem Quotienten der Shunt- und der Betriebsspannung ergibt.

Mit schwindender Batterieladung sinkt also nicht nur die Leistungsfähigkeit des Antriebs, sondern gleichzeitig steigt auch die Empfindlichkeit der propriozeptiven Motorsensoren an. Dies bedeutet, dass dieselben externen Belastungen bei schwacher Eigenenergie als intensiver wahrgenommen werden – ein Zusammenhang, der in dieser Weise auch in biologischen Systemen vorkommt.

Dies zeigt warum ein durchdachtes Schaltungsdesign und eine minimalistische Herangehensweise beim praktischen Aufbau von autonomen Robotern so wichtig sind. Die klassische Herangehensweise eines Ingenieurs wäre gewesen, zunächst den Einfluss der Betriebsspannung auf die Sensorik weitestgehend auszuschalten. Der sinnvolle, sich eigentlich natürlich ergebende Zusammenhang, müsste dann per Software im Mikrocontroller simuliert werden. Ein doppelter Mehraufwand ohne erkennbaren Mehrwert.

10.2 Do:Little

Die Ähnlichkeit von Do:Little und Fahrender Platine ist unverkennbar, trotzdem liegt dem Do:Little ein vollständig überarbeitetes Konzept zugrunde. Insbesondere wurden Erkenntnisse aus den Experimentalaufbauten zur Bewegungswahrnehmung (Kapitel 9.3 auf Seite 94) und zur Phonotaxis (Kapitel 9.4 auf Seite 97) beim Design berücksichtigt.

Wegen seiner akustischen Kommunikationsfähigkeit und da er als kommerzielles Produkt angeboten wird, erhielt der Do:Little seinen Namen in Anlehnung an Eliza Doolittle aus George Bernard Shaws Pygmalion. Abbildung 10.5 auf der nächsten Seite zeigt einige Do:Littles aus der ersten Produktionsserie. In der Großaufnahme des Do:Little kann man die im Blockschaltbild (Abbildung 10.6) angegebenen Sensoren und Aktuatoren gut identifizieren.

Prozessor und Energiehaushalt

Als zentrale Steuerung wird beim Do:Little ein 16-Bit-CISC-Mikrocontroller mit DSP-Funktionalität eingesetzt. Dieser kann variabel bis zu 20MHz getak-

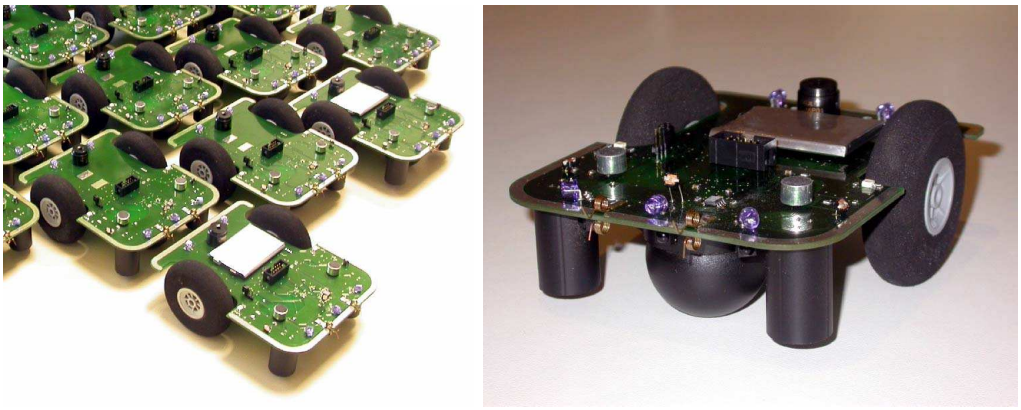

Abbildung 10.5: *(links) Erste Produktionsserie von Do:Littles – die beiden vordersten sind bereits mit Akkus bestückt. (rechts) In der Nahaufnahme erkennt man gut die verschiedenen Sensoren und die Federkontakte zur Stromaufnahme an der Vorderseite.*

tet werden, wodurch trotz der hohen Anzahl von Sensoren und Steuerungsaufgaben noch genug Rechenleistung zur Verfügung steht, um auch größere neuronale Netze mit bis zu 50 Neuronen betreiben zu können.

Der Mikrocontroller besitzt einen weiten Betriebsspannungsbereich von

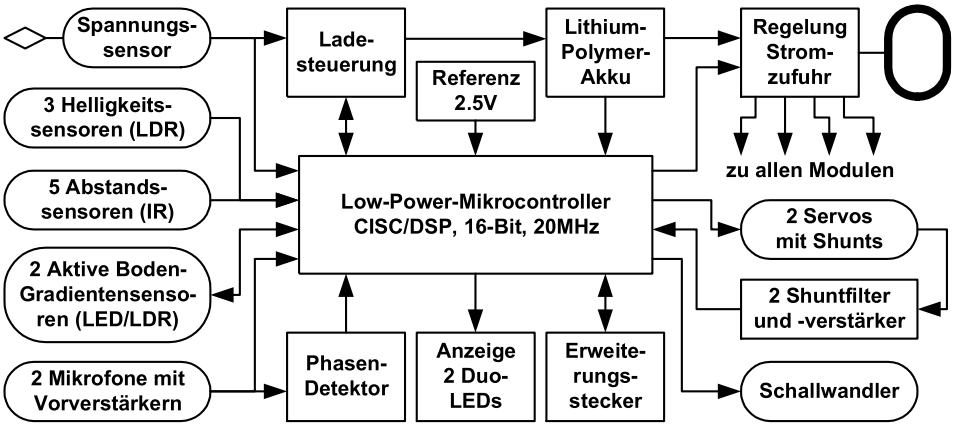

Abbildung 10.6: *Blockschaltbild des Do:Little.*

2.7–5.5V und verschiedene Betriebsmodi mit sehr geringer Leistungsaufnahme, bei denen der Stromverbrauch bis auf 0.7μA herunterfällt. Dies liegt weit unter der Selbstentladung des Akkus, weshalb der Do:Little keinen Ein-/Ausschalter besitzt, sondern über ein akustisches Wecksignal aus dem Schlaf geholt wird. Dies ist insbesondere von Vorteil, wenn eine größere Schar von Do:Littles gleichzeitig belebt werden soll, da nicht jeder einzeln in die Hand

genommen werden muss.

Den Energiespeicher bildet ein Lithium-Polymer-Akku, der einer aufwendigen Laderegelung bedarf, da in Abhängigkeit der Klemmenspannung von Konstantstrom- und Konstantspannungsladung umgeschaltet werden muss. Dies wird jedoch durch die Vorteile gegenüber anderen Akkutechnologien mehr als ausgeglichen:

- Hohe Energiedichte und kompakte Bauform
- Extrem geringe Selbstentladung
- Kein Memory-Effekt
- Dauer-/Spitzenstrom von 10/15A
- Spannungsbereich von 3.0–4.2V

In diesem Spannungsbereich lässt sich nicht nur der Prozessor, sondern auch die gesamte Elektronik inklusive der Servos betreiben. Durch die Hochstromfestigkeit kann ein Do:Little problemlos gleichzeitig Energie an mehrere andere abgeben. Der eingesetzte Akku hat eine Kapazität von 850mAh, was dem Do:Little Wachphasen von mehreren Stunden ermöglicht.

Gustatorische und propriozeptive Sensoren

Wie die Großaufnahme des Do:Little in Abbildung 10.5 zeigt, besitzt er an seiner Vorderseite bronzefarbene Federkontakte und rund um seinen Körper silberfarbene Kontaktbahnen. Diese ermöglichen den Energietransfer zwischen einzelnen Do:Littles oder von einer fremden Quelle, wie zum Beispiel einer Batterie.

Neben der Referenzspannungsquelle von 2.5V, mit deren Hilfe man die Batteriespannung als propriozeptiven Sensorwert bereitstellen kann (siehe hierzu die Beschreibung bei der Fahrenden Platine in Kapitel 10.1 auf Seite 102), ist auch die Spannung an den Federkontakten als gustatorischer Sensorwert vorhanden.

Mit diesen Informationen kann ein neuronales Netz entscheiden, ob der Do:Little Energie aufnehmen soll bzw. ob er bereit ist, eigene Energie abzugeben – er kann nämlich über ein Aktuator-Neuron seinen silberfarbenen Ring von der Energie abkoppeln. Dies kann zum Beispiel zur Selbsterhaltung notwendig sein, denn wenn sein Akku unter 3.0V fällt, wird der Akku tiefentladen.

Kern der Laderegelung ist ein diskret aufgebauter und vom Mikrocontroller gesteuerter Buck-and-Boost-Konverter, der es dem Do:Little ermöglicht Spannungen von 1–13V zu konsumieren. Theoretisch könnte also ein Do:Little einen anderen bis zuletzt aussaugen, falls dieser selbstlos genug ist. Außerdem können selbst verbrauchte 1.5V-Batterien somit noch als „Leckerli“ dienen.

Ebenfalls zur propriozeptiven Sensorik gehören die aufbereiteten Servo-Shuntspannungen, welche bereits bei der Fahrenden Platine in Kapitel 10.1 auf Seite 103 detailliert beschrieben wurden. Beim Do:Little ist lediglich die Eckfrequenz des Tiefpassfilter auf $f_0 = 19\text{Hz}$ angehoben, damit der Do:Little schneller auf externe Einflüsse reagieren kann. Die Zeitkonstante beträgt nun nur noch $T = 52\text{ms}$, was ungefähr die Hälfte unter der menschlichen Reaktionszeit liegt.

Umweltsensoren

Der Do:Little besitzt drei Helligkeitssensoren, welche an seiner Vorderseite links, mittig und rechts nach schräg oben gerichtet sind.

Fünf diskret aufgebaute Abstandssensoren – drei nach vorne, zwei nach hinten – dienen der Hinderniserkennung. Jeder Sensor besteht aus einer Infrarotsendediode (bläulich-violettes Gehäuse, siehe Abbildung 10.5 auf Seite 106) und einem zugehörigen Empfangsmodul. Die Sende- und Empfangskombinationen sind so angeordnet, dass sich vorne und hinten eine lückenlose 180°-Abdeckung ergibt. Durch eine geeignete Modulation des Sendersignals wird der Abstand zum Hindernis detektiert.

Die Abstandssensoren werden alle 100ms für wenige Millisekunden aktiv. Während der restlichen Zeit werden jedoch die Infrarotempfänger weiterhin abgefragt und können so andere Do:Littles erkennen, die in ihrer Nähe sind. Diese Information steht somit zusätzlich und getrennt von der Hinderniserkennung zur Verfügung.

Da alle Do:Littles quarzgetaktet sind, stehen die Zeitabläufe mehrerer Do:Littles stets in festen Phasenverhältnissen. Damit die eben beschriebene gegenseitige Erkennung trotzdem zuverlässig funktioniert und nicht etwa zwei Do:Littles für immer gleichzeitig ihre Abstandssensoren aktivieren, werden die Aktivphasen in gewissen Grenzen auf zufällige Weise zeitlich verschoben. So stören sich die Abstandssensoren höchstens gelegentlich, was nicht weiter ins Gewicht fällt.

Gänzlich neu entwickelt wurden die auf den Boden gerichteten aktiven Gradientensensoren. Jeder dieser Sensoren besteht aus einer roten, hocheffizienten Leuchtdiode und einem schnell ansprechenden Fotowiderstand, die zusammen in eine schwarze Abschirmung gekapselt sind. Abbildung 10.7 zeigt die Anordnung mit zurückgezogener und noch unlackierter Abschirmung.

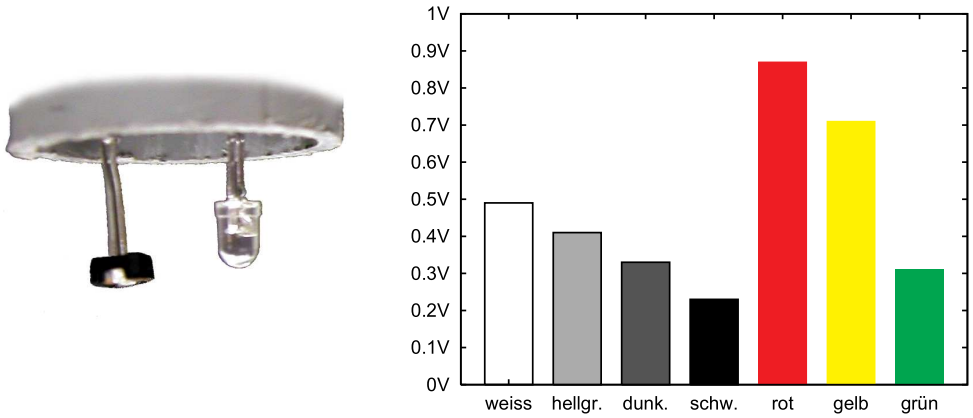

Abbildung 10.7: *(links) Anordnung der LED/LDR-Kombination im aktiven Gradientensensor. (rechts) Spannungshub am LDR bei getakteter LED in Abhängigkeit der Bodenhelligkeit bzw. -farbe.*

Betrachtet man den Spannungshub am Fotowiderstand bei getakteter Leuchtdiode, so erkennt man eine nahezu perfekt lineare Abhängigkeit vom Bodengradienten. Es mag zunächst verwundern, warum Rot einen höheren Hub hat als Weiss. Dies liegt jedoch einfach daran, dass der Fotowiderstand in einem deutlich breitbandigerem Lichtspektrum empfindlich ist, so dass sich bei ausgeschalteter Leuchtdiode und rotem Boden eine geringere Aussteuerung des Fotowiderstandes und damit ein insgesamt größerer Hub ergibt.

Durch die aktive Funktionsweise erkennt der Gradientensensor die Bodenhelligkeit unabhängig vom Umgebungslicht. Durchgeführte Tests in Dunkelheit, bei Kunstlicht und in grellem Sonnenlicht haben dies bestätigt. Damit kann der Do:Little sich an Gradientenfeldern auf dem Boden orientieren, selbst wenn sich stark ausgeleuchtete Zonen mit Schlagschatten abwechseln.

Über die erkannte Bodenhelligkeit und den absoluten Spannungswert des Fotowiderstandes bei ausgeschalteter Leuchtdiode ist es umgekehrt auch möglich, Schattengebiete unabhängig von der Bodenbeschaffenheit zu meiden bzw. aufzusuchen. Zusammen mit den nach oben gerichteten Helligkeitssensoren ergeben sich hier differenzierte Wahrnehmungsmöglichkeiten.

Wie man auf der rechten Hälfte von Abbildung 10.5 auf Seite 106 sieht, sind die fertigen Sensoren links und rechts unter der Vorderseite platziert.

Diese Positionierung ermöglicht es, bei voller Geradeausfahrt auf einem Tisch den Abgrund zu erkennen und noch rechtzeitig abzudrehen.

Akustische Kommunikation

Wie eingangs erwähnt, wurde der Do:Little wegen seiner akustischen Kommunikationsfähigkeit so getauft. Diese verdankt er einer Kombination von präzise aufeinander abgestimmten Komponenten und signalverarbeitenden Technologien.

Oberstes Designziel war es, eine möglichst störsichere Gesamtkommunikationsstrecke sicherzustellen, die trotzdem gut im menschlichen Hörbereich liegt, so dass ein Betrachter die Kommunikationssignale mithören kann. Andererseits ist es auch wünschenswert, einem größeren Publikum eine Horde Do:Littles in Aktion demonstrieren zu können – klassischerweise in einer Halle (der Begriff muss hier wörtlich genommen werden), die meist mit Stimmengewirr gefüllt ist. Ein Do:Little soll erkennen, welches Signal aus welcher Richtung kommt, auch wenn er gerade umherfährt.

Um dies alles zu leisten, wurden zunächst verschiedene Servotypen auf ihre Geräuschentwicklung im Leerlauf und bei Belastung untersucht, jeweils über den Spannungsbereich von 3.0–4.2V. Da die Mikrofonkapseln mechanisch auf dem Do:Little befestigt sind, werden die Servogeräusche als Körperschall auf die Mikrofone übertragen und erreichen höhere Pegel, als die aus der Luft empfangenen Schallwellen. Abbildung 10.8 zeigt einen 1s dauernden Ausschnitt des induzierten Mikrofonsignals (blau) und das zugehörige Frequenzspektrum (rot). Um eine gute Frequenzauflösung zu erhalten, wurde das Signal zuvor mit einem Blackman-Harris-Fenster gewichtet.

Kommunikationssignale sind am störsichersten, wenn sie in einem geräuscharmen Frequenzbereich untergebracht werden. Beim verwendeten Servo bietet sich hier der Bereich von $f = 0.50$–1.10kHz an. Rechnet man mit einer Schallgeschwindigkeit in der Luft von $c = 340$m/s, so beträgt die Wellenlänge in diesem Frequenzbereich $\lambda = c/f = 30.9$–68.0cm, was leider zu groß ist – wie weiter unten noch erläutert wird.

Das erste interessante Tal befindet sich bei $f = 2.00$–2.10kHz. Daher wurde als nächstes der Frequenzgang von 14 Schallwandlern unterschiedlicher Bauform und mit unterschiedlichen Wandlerprinzipien ausgemessen. Hierzu wurden die Schallwandler mit einem 10s dauernden Frequenzsweep angesteuert und das Signal in 1m axialer Entfernung von der Schallöffnung ausgewertet. Das Ergebnis ist in grau dargestellt und so skaliert, dass gleiche Frequenzen übereinander liegen. Der gefundene Schallwandler hat seine Resonanzfrequenz bei $f_r = 2.05$kHz und passt damit exakt in den vorgegebenen Bereich.

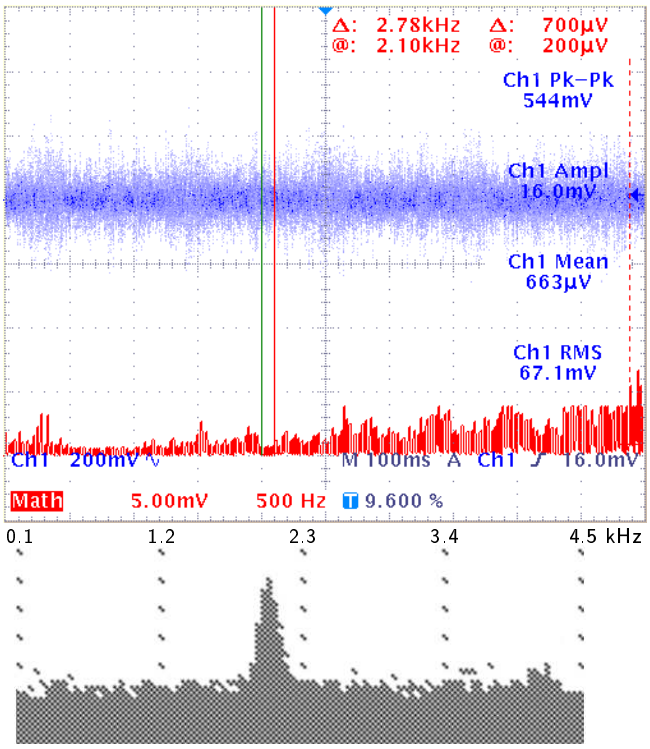

Abbildung 10.8: *(oben) Mit der Mikrofonkapsel aufgenommenes Geräusch der beiden Servos (blau) und zugehöriges Frequenzspektrum (rot) nach Gewichtung mit Blackman-Harris-Fenster. Zwischen 2.00kHz und 2.10kHz existiert ein geräuscharmer Bereich (grüne und rote senkrechte Markierung), in welchem sich das Nutzsignal unterbringen lässt. (unten) Frequenzgang des aus 14 Bauformen ausgewählten Schallwandlers, gemessen in 1m axialer Entfernung von der Schallöffnung, unter Verwendung eines 10s langen Frequenzsweeps. Die deutlich sichbare Resonanzfrequenz des Schallwandlers liegt durch die Bauart bedingt bei 2.05kHz und passt somit optimal zum Geräuschspektrum der Servos.*

Kurze, modulierte Chirpsignale auf dieser Trägerfrequenz klingen ungefähr wie Grillenzirpen und sind vom Menschen gut wahrnehmbar. Andererseits besitzen menschliche Laute in diesem Bereich nur noch wenig Energie und konkurrieren nicht mit den Signalen des Do:Littles, so dass bereits einige der geforderten Bedingungen erfüllt sind.

Menschen werten zum Richtungshören je nach Frequenzbereich im wesentlichen drei Kriterien aus:

- Amplitudendifferenz linkes/rechtes Ohr
- Phasendifferenz linkes/rechtes Ohr

- Zeitdifferenz der ersten eintreffenden Wellenfront linkes/rechtes Ohr

Wie die Experimente aus der Phonotaxis gezeigt haben, eignet sich die Amplitudendifferenz nur bei Nahbesprechung oder aufgesetzten Kopfhörern. Für die Do:Littles ist eine Kombination aus den letzten beiden Kriterien am besten geeignet. Man betrachte hierzu Abbildung 10.9. Je nach Richtung des Schalleinfalls liegt die Laufzeitdifferenz zwischen Null und einem maximalen Wert, der sich aus dem Mikrofonabstand ergibt.

Beträgt die Laufzeitdifferenz exakt $T/2 = 1/(2f)$, dann ist die Phasendifferenz $\Delta\phi = \pm\pi$ und es ist nicht mehr erkennbar, ob das Signal vom linken Mikrofon dem vom rechten voreilt oder umgekehrt. Die Laufzeitdifferenz sollte also unter $T/2$ liegen.

Dies kann man erreichen, indem man den Mikrofonabstand oder die Signalfrequenz entsprechend verringert. Andererseits wachsen damit auch die Ansprüche an die Signalverarbeitung, denn je kleiner die Phasendifferenzen sind, desto exakter muss deren Auswertung erfolgen.

Daher beträgt der Mikrofonabstand beim Do:Little 66mm, was bei der verwendeten Frequenz von $f_r = 2.05$kHz weniger als $\lambda/2 = c/(2f_r) \approx 83$mm ist. Das erlaubt noch Spielraum für höhere Frequenzen und bietet trotzdem genug Phasendifferenz, so dass der Do:Little die Signalrichtung auf wenige Grad genau bestimmen kann. Die Verwendung einer tieferen Signalfrequenz ginge zu Lasten dieser Genauigkeit.

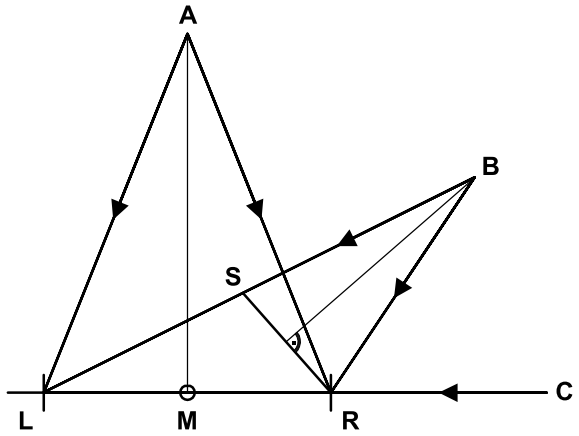

Abbildung 10.9: *Richtungsabhängiger Schalleinfall in das linke (L) und rechte (R) Mikrofon. Aus Position A gibt es keine Laufzeitdifferenz zwischen Links und Rechts. Aus Position B ist die Laufzeitdifferenz proportional zur Strecke $\overline{SL}$, aus Position C entspricht sie der Strecke $\overline{RL}$ und ist daher maximal.*

Erweiterungen

Der Do:Little besitzt einen Anschluss, auf den Erweiterungsmodule gesteckt werden können. Folgende Module wurden bisher entwickelt:

- Power: Erweiterung der Akkukapazität auf das Dreifache
- Vision: Kameramodul, welches die in Kapitel 9.3 beschriebene Bewegungswahrnehmung ermöglicht
- Sound: Digitaler Signalprozessor mit Endstufe und hochwertigem Lautsprecher, der die Aktivierung interner Neuronen hörbar machen kann
- Memory: Schnittstelle zu einer CompactFlash Speicherkarte, auf der zur Laufzeit alle neuronalen Signale für eine spätere Analyse gespeichert werden

Diese und weitere, noch in Entwicklung befindliche Module, werden automatisch vom Do:Little erkannt und die Firmware stellt intern unmittelbar entsprechende Sensor- und Aktuatorschnittstellen zur Verfügung. Die Erweiterungsmodule lassen sich problemlos kombinieren, indem man sie gestapelt aufsteckt.

Kapitel 11

Laufmaschinen

Während bei radgetriebenen Robotern die ausgefeilte Sensorik den Entwicklungsschwerpunkt bestimmt, so ist dies bei Laufmaschinen stets die Morphologie und motorische Ansteuerung der Beine.

Beim Aufbau der im folgenden vorgestellten Systeme stand zunächst die Frage im Vordergrund, auf welche Weise das Laufverhalten von Beinform, Fußbeschaffenheit, Schwerpunkt sowie Anzahl und Anordnung der Freiheitsgrade bestimmt wird.

Dies hängt natürlich eng mit der verwendeten Ansteuerung der Freiheitsgrade zusammen. Daher wurde der in Kapitel 4.1 auf Seite 35 beschriebene SO(2)-Oszillator eingesetzt, dessen Ausgangssignale sich in weiten Bereichen variieren lassen. Dabei stellte sich einerseits heraus, dass Laufrichtung und -geschwindigkeit gut über einzelne Netzwerkparameter steuerbar sind – andererseits zeigte sich aber deutlich, dass morphologische Kriterien das Laufverhalten viel stärker beeinflussen.

Die Laufmaschine Oktavio (siehe Kapitel 11.4) setzt sich von den anderen Systemen in vielerlei Hinsicht ab. Seine Beine sind vollständig autark und energetisch genauso durchdacht wie der Do:Little (siehe hierzu die Beschreibung in Kapitel 10.2 auf Seite 107). Darüberhinaus ist Oktavio als universelle Experimentierplattform konzipiert, die zusätzliche Sensoren und weitere Nutzlast aufnehmen kann.

11.1 Lucy

Als Argument gegen den Bau von Laufmaschinen wird oftmals angeführt, dass diese gegenüber fahrenden Robotern erheblich mehr Freiheitsgrade besitzen müssen und daher entsprechend schwieriger zu steuern wären. Dass dies nicht zwangsläufig der Fall sein muss, zeigt der Aufbau von Lucy in

Abbildung 11.1. Lucy besitzt ein vorderes und ein hinteres Beinpaar aus ummanteltem Federstahl, welches jeweils drehend an einem Servo befestigt ist.

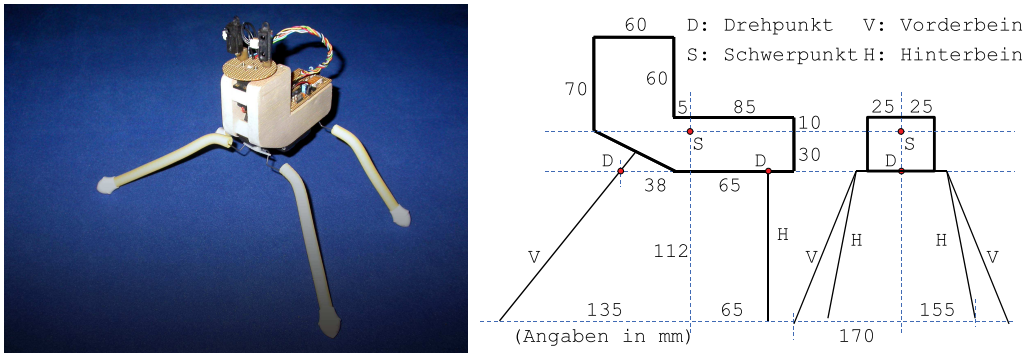

Abbildung 11.1: *(links) Lucy ist mit einem drehbaren Kopf ausgestattet und erreicht mit nur einem Freiheitsgrad je Beinpaar eine hohe Laufgeschwindigkeit. (rechts) In der Skizze sieht man die Lage des Schwerpunkts (S) und die notwendige Neigung des vorderen Beinpaars (V).*

Dreht sich der Servo des hinteren Beinpaars von oben betrachtet im Uhrzeigersinn, dann bewegt sich das linke Bein nach vorne und das rechte nach hinten – analog bei entgegengesetzter Drehrichtung.

Die Mittelachse des vorderen Beinpaars ist, wie in Abbildung 11.1 von der Seite skizziert, um ungefähr 45° aus der senkrechten Mittelachse nach hinten geneigt. Neben der für das hintere Paar beschriebenen Drehbewegung, hebt sich daher beim vorderen Paar gleichzeitig das vorwärtsstrebende Bein nach oben an.

Lucy neigt sich wegen ihrer Masse von $m = 0.31$kg leicht nach vorne und durch die Elastizität des Federstahls bleiben alle vier Füße am Boden, pressen jedoch mit unterschiedlicher Kraft auf den Untergrund. Deswegen rutschen zwei diagonal gegenüberliegende Füße nach vorne, während die anderen beiden auf der Stelle verweilen.

Ab einer gewissen Laufgeschwindigkeit überlagern sich zunehmend die dynamischen Effekte des Federstahls, was sich positiv auf die Laufgeschwindigkeit und Gesamtästhetik auswirkt.

Funktionsmodule

Die Schaltungsdetails von Lucy sind fast identisch mit denen der Fahrenden Platine (Kaptiel 10.1), so dass es genügt, hier anhand des Blockschaltbilds in Abbildung 11.2 kurz die Unterschiede darzustellen.

Die paarig angeordneten Helligkeits- und Abstandsensoren sind auf Lucys drehbaren Kopf montiert. Dadurch lassen sich Experimente mit aktiv bewegter Sensorik durchführen. Für die Kopfdrehung wird ein zusätzlicher dritter Servo eingesetzt, jedoch ohne Shunt-Widerstand und Filter.

Neu hinzugekommen ist ein Zwei-Achsen-Beschleunigungssensor, der statische und dynamische Beschleunigungen in den Richtungen vor/zurück und hoch/runter erfasst. Über die Erdbeschleunigung kann somit auch Lucys Saggitalneigung berechnet werden. Die Auswertung der Neigung und Beschleu-

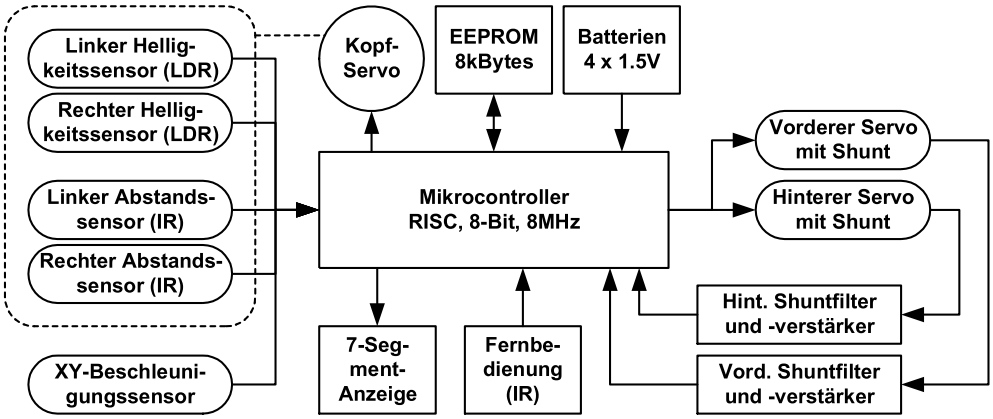

Abbildung 11.2: *Blockschaltbild von Lucy.*

nigungen bietet interessante Möglichkeiten das Laufverhalten direkt aus den Sensordaten numerisch zu bewerten, wie dies zum Beispiel zur Implementierung einer Fitnessfunktion notwendig ist, wenn man eine On-board-Evolution durchführen möchte.

Für eben diesen Zweck ist Lucy auch über eine Infrarot-Fernbedienung fernsteuerbar und mit einer 7-Segment-Anzeige ausgestattet, die den aktuellen Fitnesswert oder andere Daten ausgeben kann.

11.2 TED

Das Akronym TED steht für *Two-degrees-of-freedom Experimental Device* (Experimentalaufbau mit zwei Freiheitsgraden) und bezeichnet Lucys Nachfolger. Er funktioniert nach demselben Schema wie Lucy, ist aber in jeder Hinsicht minimal gehalten.

Wie Abbildung 11.3 zeigt, besteht er im wesentlichen nur noch aus zwei Miniaturservos, die zwischen zweimal zwei 1.5V-Batterien montiert sind. Neben den Fotowiderständen zur Erkennung von Helligkeitsunterschieden und

einer roten Leuchtdiode als Signalgeber gibt es als aktives elektronisches Bauteil lediglich einen Mikrocontroller – nicht größer als ein kleiner Fingernagel.

TED entstand primär als Anschauungs- und Experimentalobjekt. Durch seine kompakte Bauweise und geringe Masse von $m = 90$g, die fast ausschliesslich durch die Batterien bestimmt ist, lässt er sich problemlos zu Präsentationen mitnehmen, beispielsweise um die Funktionsweise des SO(2)-Oszillators (Kapitel 4.1) oder den Einfluss der Morphologie auf das Laufverhalten zu demonstrieren.

Man benötigt nur wenige, preiswerte Einzelteile, um an einem Nachmittag selbst einen TED zusammenzubauen, individuelle Beinformen zu biegen und schließlich das selbst kreierte neuronale Netz aufzuspielen und auszuprobieren (siehe Abbildung 11.4 auf der nächsten Seite). TED eignet sich daher durchaus als anspruchsvolles Unterrichtsprojekt für Kleingruppen.

Abbildung 11.3: *(links) TEDs Körper ist nur unwesentlich größer als vier 1.5V-Batterien der Größe AAA. (rechts) An die Metallbeine lassen sich zum Ausprobieren unterschiedliche Fußvarianten anschrauben. Die hier abgebildeten Füße bestehen aus hartelastischem Plastik und haben sich auf Teppichboden, Steinpflaster und Rasen gleichermaßen bewährt.*

Bemerkung zur Firmware

Das Bestreben, TED so minimalistisch wie möglich zu gestalten, wurde auch bei der Prozessorauswahl beibehalten. Der eingesetzte Mikrocontroller besitzt nur acht Anschlüsse, davon drei für Masse, Betriebsspannung und Reset, die restlichen fünf für zwei Servos, zwei Fotowiderstände und eine Leuchtdiode mit bereits integriertem Vorwiderstand.

Es handelt sich um einen 8-Bit-RISC-Mikrocontroller mit weitem Betriebsspannungsbereich von 2.7–5.5V und intern generierter Taktung von 1.6MHz, so dass das sonst übliche externe Quarz entfällt.

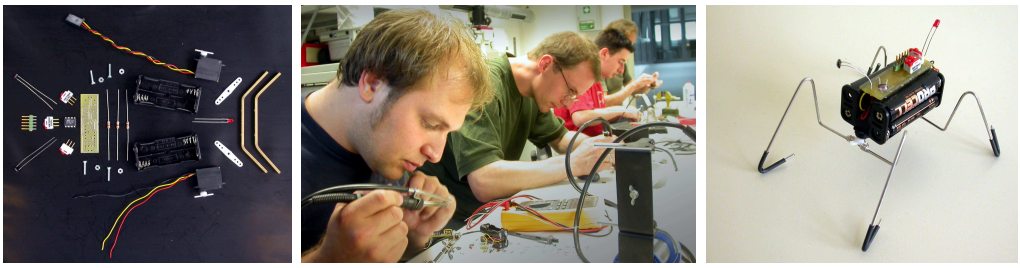

Abbildung 11.4: *(links) Ein TED besteht nur aus wenigen, preiswerten Einzelteilen. (Mitte) Der Zusammenbau ist lehrreich und erfordert Fingerspitzengefühl. (rechts) Die selbstbestimmte Beinform wirkt sich unmittelbar auf das Laufverhalten aus.*

Da der Mikrocontroller nur über 1kByte Flash-Programmspeicher verfügt und außer seinen 32 Registern mit 8-Bit Breite keinen Arbeitsspeicher besitzt, musste die Firmware vollständig in Assembler erstellt werden. Insbesondere konnte aufgrund dieser Beschränkungen auch keine Fließkomma-Arithmetik verwendet werden.

Um den neuronalen SO(2)-Oszillator ausreichend tieffrequent für Laufbewegungen betreiben zu können, ist eine 8-Bit-Auflösung für die Darstellung der Verbindungsgewichte und Ausgangssignale zu gering. Für 24-Bit genügt die Anzahl der Register nicht, daher ist die Verwendung von 16-Bit-Festkomma-Arithmetik der einzig mögliche Kompromiss.

Zur optimalen Ausnutzung der Rechengenauigkeit wurden für die Verbindungsgewichte, Ausgangssignale und Aktivierungen der simulierten Neuronen unterschiedliche Festkommaformate gemäß Tabelle 11.1 verwendet.

	Gewichte	Aktivitäten	Ausgänge
Bit-Breite	16	32	16
Vorkommastellen	4	9	1
Nachkommastellen	12	23	15
Wertebereich	$[-8, +8)$	$[-512, +512)$	$[-1, +1)$
Genauigkeit	$\approx 10^{-4}$	$\approx 10^{-7}$	$\approx 10^{-5}$

Tabelle 11.1: *Aufschlüsselung der verwendeten Bit-Breite und internen Rechengenauigkeit für die im TED simulierten neuronalen Signale.*

Die Transferfunktion *tanh* wurde, wie in Kapitel 2.2 auf Seite 17 beschrieben, unter Ausnutzung der Punktsymmetrie zum Nullpunkt über fünf lineare Segmente approximiert. Trotz der deutlich reduzierten Rechengenauigkeit und stückweise linearisierten *tanh*-Funktion schwingt sich der SO(2)-

Oszillator zuverlässig auf seinen Arbeitspunkt ein und läuft stabil, solange die Batterie reicht.

11.3 Krabbelroboter

Nach dem eingehenden Studium der Einflüsse von Steuersignal, Beinform und Fußbeschaffenheit auf das Laufverhalten, muss man konsequenterweise auch die Frage stellen, welche Rolle den Freiheitsgraden zukommen. Sieht man vom Problem der Ansteuerung ab, so bietet jeder zusätzliche Freiheitsgrad sicherlich zusätzlichen Aktionsradius. Aber kann man auch mit weniger als zwei Freiheitsgraden auskommen?

Die Antwort lautet: ja – sowohl bei radgetriebenen Robotern, wie auch bei Laufmaschinen. Für radgetriebene Roboter verwendet man einen differentiellen Antrieb, steuert aber nur das linke Rad an. Das rechte Rad koppelt man mechanisch so an das linke, dass es sich unabhängig von der Bewegungsrichtung des linken Rads stets vorwärts bewegt. Dies lässt sich auf einfache Weise mit zwei gegenläufigen Rutschkupplungen erreichen. Damit kann man entweder geradeaus fahren, oder sich linksherum auf der Stelle drehen. Man findet so etwas gelegentlich bei einfachen Kinderspielzeugen.

Laufmaschinen kann man auch mit aufwendiger Mechanik ausstatten und eine einzige rotatorische Bewegung so in sechs Beine umlenken, dass vorwärtslaufen und sich drehen daraus resultieren. Interessanterweise kann man aber auch noch vorwärts kommen, wenn man fordert, dass der Freiheitsgrad die Beine mechanisch direkt ansteuert. Die Laufmaschine degeneriert dann zum Krabbelroboter, wie in Abbildung 11.5 dargestellt.

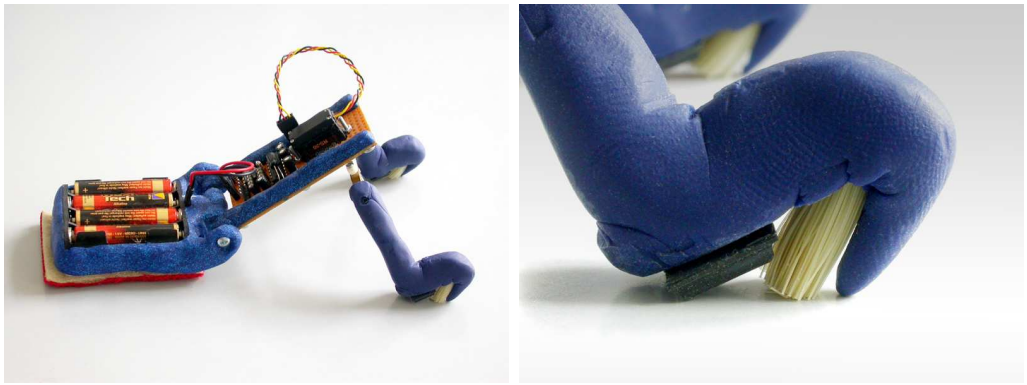

Abbildung 11.5: *(links) Der Krabbelroboter besitzt nur einen Freiheitsgrad und kann sich trotzdem forwärts bewegen. (rechts) An den Füßen des Krabbelroboters wurden Materialien eingesetzt, die stark richtungsabhängige Haftreibungen besitzen.*

Der Trick liegt in der Verwendung geeigneter Materialien für alle Stellen, die Bodenkontakt haben. Neben den beiden Vorderfüßen ist dies der hintere Körperteil, auf welchem der Krabbelroboter vorwärtsrutscht. Am besten bewährt hat sich an dieser Stelle die Bespannung einer Fusselbürste, welche in einer Richtung fast reibungslos über das Material gleitet, in der entgegengesetzten Richtung aber kleinste Borsten aufrichtet und dadurch eine hohe Haftreibung besitzt. Schlangenhäute funktionieren teilweise auf dieselbe Weise, nur dass sich dort Schuppen anstelle von Borsten aufrichten.

Für die beiden Vorderfüße wurde eine Kombination aus Krallenform, nach hinten gerichteter Dachshaarborste und Gummifuß verwendet. Die einzelnen Elemente greifen jeweils optimal bei großschlaufigem Teppichboden, feinstrukturierten Stoff- oder Steinoberflächen und glatten Resopal- oder Glasoberflächen.

Der Krabbelroboter kann sich nur vorwärts bewegen, aber leichte Kurven nach links und rechts sind dennoch möglich, indem der symmetrischen Schwingbewegung des vorderen Beinpaars ein kleiner Offset in die entsprechende Richtung überlagert wird (siehe hierzu auch Kapitel 4.1 auf Seite 37).

11.4 Oktavio

Die in Abbildung 11.6 gezeigte Laufmaschine Oktavio ist das aufwändigste der hier vorgestellten Systeme. Oktavio besteht aus einem Rumpf und acht identisch aufgebauten Beinen, die per Schnappverschluss am Rumpf befestigt werden.

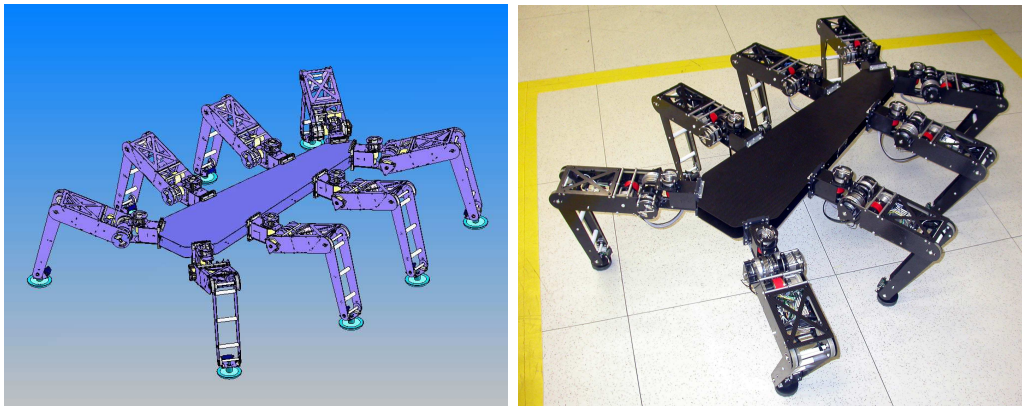

Abbildung 11.6: *(links) Konstruktionszeichnung des Oktavio in perspektivischer Ansicht. (rechts) Fertig aufgebautes System Ende November 2004.*

Oktavio entstand als universelle Testplattform für neuronale Laufmaschi-

nensteuerungen und Verbundsysteme, die den Rumpf als Sensorträger verwenden. Der Konstruktion lagen folgende Designkriterien zugrunde:

- Größe über alles: 150cm lang, 100cm breit, 30cm hoch. Soll auch in Grünanlagen im hohen Gras laufen können.
- Modularer Aufbau, Beine müssen schnell und einfach umgesteckt werden können. Inter-Bein-Kommunikation muss sofort und ohne Reset funktionieren.
- Vollständige Autonomie der einzelnen Beine, Stromversorgung und Rechenleistung müssen im Bein enthalten sein. Kein Ein-/Ausschalter – das Bein gelangt über äussere Sensorstimulation reflexartig aus dem Ruhezustand.
- Effizienter Energiehaushalt, möglichst geringe Masse der Beine und minimalste Stromaufnahme im Stand. Laufzeit im Bereich von einer Stunde. Gleichzeitiges Aufladen aller Beine am Rumpf möglich.
- Hochaufgelöste Fußsensorik, ähnlich einem menschlichen Fußballen.
- Störsichere Elektronik, verschleiss- und wartungsfreie Konstruktion.
- Möglichst kostengünstige Realisierung.

Der erste Aufbau des Oktavio entstand aus diesem Kriterienkatalog innerhalb von etwa eineinhalb Jahren.

Beinaufbau

Der Rumpf bestimmt im wesentlichen die Körperform von Oktavio und enthält die Schnappverschlüsse, mit denen die Beine am Rumpf verbunden werden. Außer der Verkabelung des Beinbusses (siehe weiter unten) ist im Rumpf nichts weiter vorhanden, so dass es genügt ein einzelnes Bein zu betrachten (siehe hierzu Abbildung 11.6).

Oktavios Bein besitzt drei Gelenke, die von Getriebemotoren angesteuert werden. Alle Motoren sind über Drehmomentkupplungen mit den Gelenken verbunden, so dass externe Stöße oder andere Fremdbelastungen die Motorgetriebe nicht zerstören.

Vom Rumpfflansch an kommt zunächst eine senkrecht stehende Gelenkachse, welche für die Vor-/Zurückbewegung des Beins zuständig ist. Unmittelbar dahinter liegt die waagrechte Gelenkachse für das Anheben und Abdrücken am Boden. Sie ist mit einer einstellbaren Vorspannfeder versehen, so

dass die Höhe des Rumpfes in Ruhelage definiert werden kann. So kann auch das Gewicht von Zuladungen ausgeglichen werden, wie zum Beispiel einer auf dem Rumpf montierten Kamera.

Die dritte aktiv getriebene Achse ist das Kniegelenk. Der zugehörige Motor ist jedoch bei den anderen beiden in Rumpfnähe eingebaut, um eine optimale Gewichtsverteilung im Bein zu haben. Wie bei biologischen Systemen ist das Bein am Oberschenkel schwerer und wird zum Fuß hin leichter. Die Masse des Beins beträgt $m = 1.7\text{kg}$.

Schliesslich gibt es noch ein Fußgelenk, das über Getrieberiemen so ausgerichtet wird, dass der Fuß bei waagrechtem Rumpf unabhängig von der Beinstellung stets senkrecht zum Boden steht. Zusätzlich bringt eine Rückholfeder den senkrecht frei drehbaren Fuß stets wieder in eine definierte Lage zurück, wenn sich der Fuß in der Luft befindet. Dies ist für die korrekte Funktion des Fußsensors notwendig.

Die vollständige Elektronik ist, wie in Abbildung 11.7 gezeigt, auf zwei Platinen untergebracht, welche exakt an die mechanische Bauform des Beins angepasst sind.

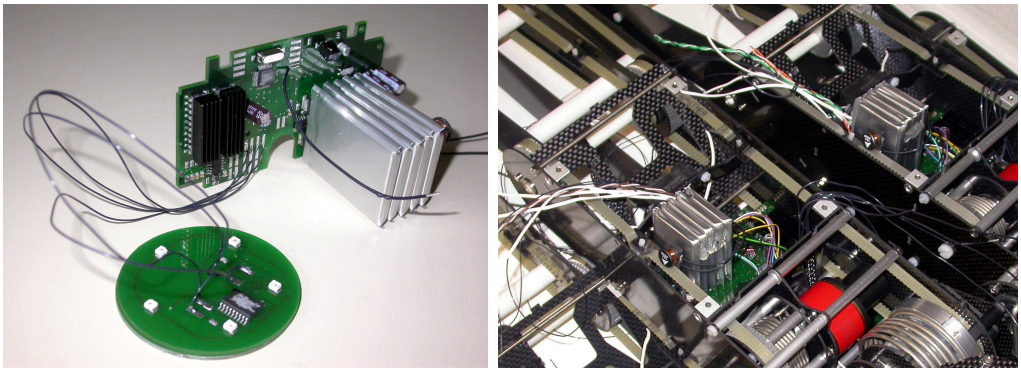

Abbildung 11.7: *(links) Kreisförmige Platine des Fußsensors und Hauptplatine des Beins. Gut erkennt man den silbernen Akkublock und die beiden schwarzen Kühlkörper der H-Brücken. (rechts) Die teilverkabelte Hauptplatine liegt zum weiteren Einbau im Elektronikfach.*

Auf einer kreisrunden im Fuß eingebauten Platine befindet sich das Auswertungsmodul des Fußsensors. Den vollständigen Rest der Elektronik bildet eine mit SMD und bedrahteten Bauteilen beidseitig gemischt bestückte Platine. Sie ist in einem speziellen Fach zwischen den Antriebsriemen bei den Getriebemotoren untergebracht (vergleiche hierzu Abbildung 11.6 auf Seite 120).

Der Deckel des Elektronikfachs besitzt große Aussparungen, damit der eingebaute und nach schräg oben gerichtete Infrarotempfänger die Signale

einer Fernbedienung detektieren kann. Wie Tests ergaben, lässt sich Oktavio zuverlässig aus über fünf Metern Entfernung fernsteuern.

Beinkommunikation und Energiehaushalt

Einen Überblick aller sensomotorischen und energetischen Fähigkeiten des Beins gibt das Blockschaltbild in Abbildung 11.8.

Das Bein ist über einen kodierten Anschluss, der in den Schnappverschluss mit eingearbeitet wurde, mit dem Beinbus verbunden. Über den Beinbus läuft sämtliche Kommunikation zwischen den Beinen und optional auch zu einem am Rumpf angeschlossenen PC. Es handelt sich um einen symmetrischen RS-485-Bus, der zur Erhöhung der Störsicherheit mit reduzierter Flankensteilheit betrieben wird. Die einzelnen Beine schalten im Token-Ring-Verfahren zwischen Slave- und Master-Betrieb hin und her. Die RS-485-Schnittstelle des Beins stellt sicher, dass im Slave-Betrieb nicht fälschlicherweise ein Signal eingelesen wird, wenn die beiden Busleitungen auf gleichem Spannungspegel liegen. Dies kann passieren, wenn im laufenden Betrieb Beine abgezogen werden.

Rumpfseitig ist jeder Beinanschluss so kodiert, dass ein angestecktes Bein weiß, ob es sich gerade an der rechten oder linken Körperhälfte sowie vorne, mittig oder hinten befindet. Außerdem lassen sich über den Beinbus bequem alle Beine gleichzeitig aufladen.

Als Energiespeicher dienen fünf Lithium-Polymer-Akkus, deren Vorteile bereits beim Do:Little (siehe hierzu Kapitel 10.2 auf Seite 107) beschrieben wurden. Die Energiesteuerung des Oktaviobeins ist jedoch deutlich komplexer aufgebaut als beim Do:Little, da beim Ladevorgang seriell geschalteter Lithium-Polymer-Zellen ständig Ladungsausgleiche vorgenommen werden müssen. Ansonsten würde womöglich eine der Zellen überladen und könnte im schlimmsten Fall explodieren.

Die zur Verfügung stehende Energie ist beachtlich. Die Klemmenspannung des Akkublocks liegt bei 15–21V. Da die Zellen problemlos kurzzeitig 15A liefern, können bis zu 315W rekrutiert werden – bei acht Beinen bedeutet dies 2,5kW für den gesamten Oktavio!

Glücklicherweise benötigen die Motoren nur deutlich geringere Ströme, so dass die 0.13kWh eines vollgeladenen Oktavios ungefähr für eine Stunde Umherlaufen reichen.

Wenn die Motoren nicht angesteuert werden, können sie sich, je nach neuronaler Steuerung, im Leerlauf oder im Bremszustand befinden. In diesen beiden Fällen geben die Motoren Energie ab, wenn das Bein von außen bewegt wird. Diese wird dem Akkublock zugeführt und reicht völlig aus, um den Mikrocontroller aus dem Ruhezustand zu holen.

Sensomotorik

Zur Motoransteuerung werden drei H-Brücken mit Shuntwiderständen eingesetzt, so dass detektiert werden kann, ob sich der gewünschten Beinbewegung äußere Widerstände entgegensetzen. Nebenbei kann über die verstärkte Shuntspannung während eines Selbsttests auch ein möglicher Wicklungsschluss oder ein durchgebrannter Motor festgestellt werden. Wegen der

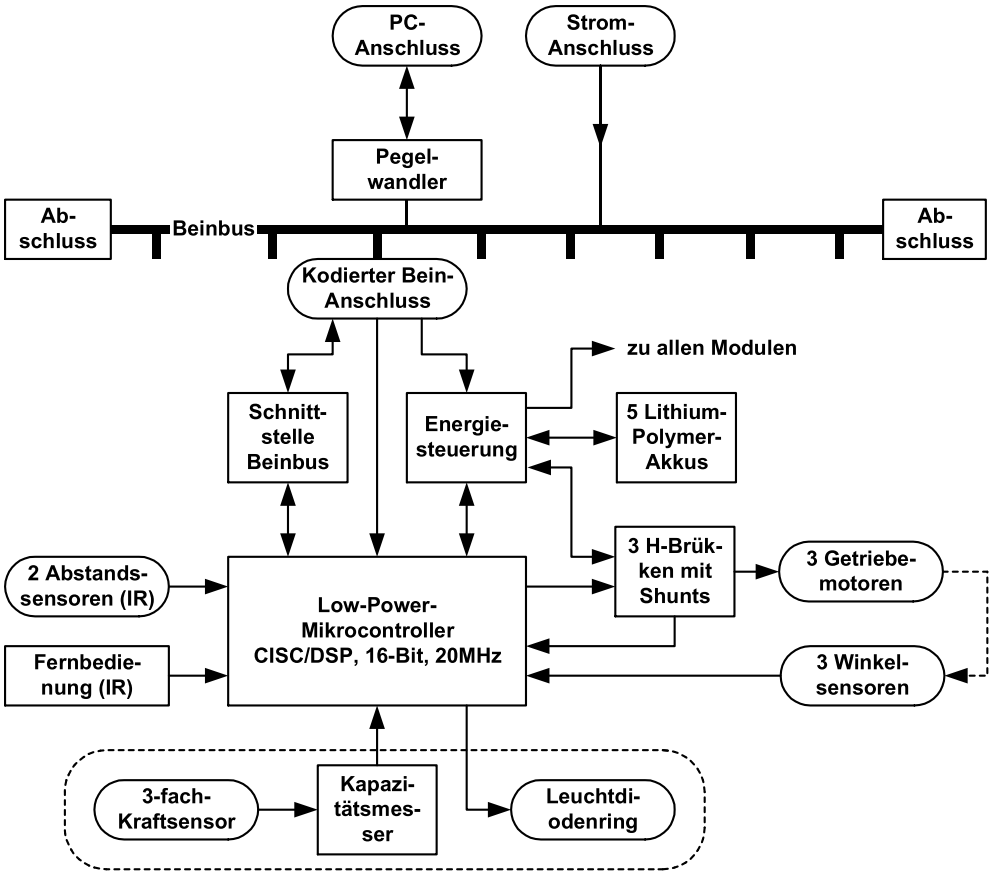

Abbildung 11.8: *Blockschaltbild des Oktavio.*

weiter oben erwähnten Drehmomentkupplung arbeiten die Motoren jedoch stets innerhalb ihrer zulässigen Grenzen. Neben der Auswertung der Shuntspannungen gibt es noch eine zweite propriozeptive Rückkopplung aus den Gelenken. Über flexible Dehnungssensoren, die mechanisch in die Gelenkkonstruktionen mit integriert wurden, erhält Oktavio Informationen über die aktuellen Winkelstellungen seiner Gelenke. Die Dehnungssensoren bestehen aus einer beidseitig in Plastikstreifen gefassten, länglich aufgedampften Widerstandsschicht, die bei Abwinklung des Sensors gedehnt wird und da-

durch ihren Widerstand erhöht. Im Gegensatz zur Shuntsensorik lassen sich die Winkelsensoren auch im passiven Betriebsmodus sinnvoll auswerten und geben sowohl bei aktiver wie auch nicht aktiver Motoransteuerung Auskunft über äußere Einflüsse.

Als Umweltsensoren fungieren zwei Infrarot-Abstandssensoren, von denen einer hinter dem Kniegelenk sitzt und parallel zum Boden vom Rumpf weggerichtet nach aussen zeigt. Der zweite sitzt knapp über dem Fuß und detektiert bei gehobenem Bein den Abstand zum Boden.

Fußaufbau

Wie weiter oben bereits erwähnt, stellt die Beinmechanik sicher, dass der Fuß bei waagrechtem Rumpf stets senkrecht zum Boden auftritt. Wenn dies geschieht, liefert der Fußsensor Informationen über die Kraft, mit welcher der Fuß auf den Boden drückt und über die genaue Lage des Aufdruckpunkts. Dies erfolgt relativ zum Rumpf, das heißt es wird stets angegeben, wie weit der Fuß gerade innen/aussen sowie vorne/hinten auftritt. Hierzu wird die rumpfseitige Kodierung des Beinanschlusses mit herangezogen.

Der Fuß ist vollständig gekapselt und wird mit einer Schraube am Bein befestigt, aus deren Zentralbohrung die Anschlusskabel geführt werden. Den Aufbau und das Funktionsprinzip sieht man in Abbildung 11.9.

An der Befestigungsschraube ist eine Platine montiert, auf deren Oberseite sich die Auswertungselektronik in SMD-Bauweise befindet. Hier sind

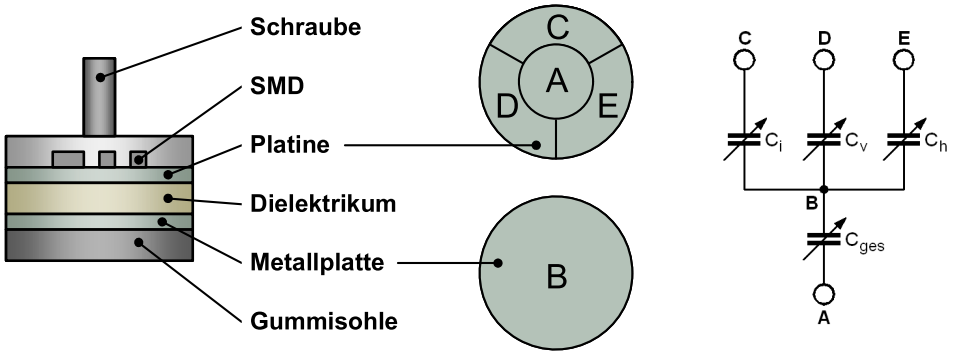

Abbildung 11.9: *(links) Der Fuß besteht aus fünf verschiedenen Schichten, von denen die mittleren drei einen knautschbaren, vierfachen Plattenkondensator bilden. (Mitte) Geometrische Anordnung der Kupferflächen. (rechts) Äquivalente elektronische Schaltung des vierfachen Kondensators. Die den Kupferflächen entsprechenden Punkte sind mit A–E gekennzeichnet.*

auch die durch die Schraube geführten Kabel sowie fünf in einem Kreis lie-

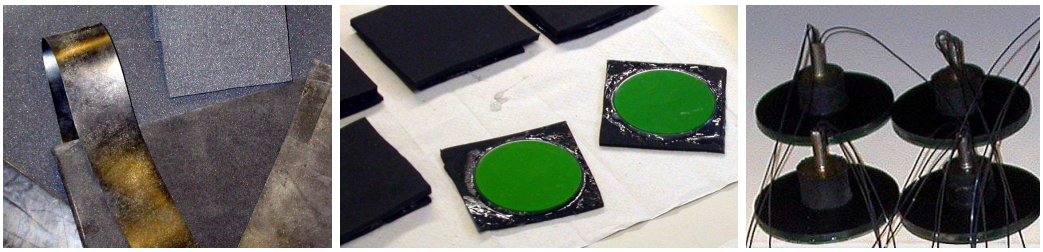

Abbildung 11.10: *(links) Einige Materialproben, deren Verwendbarkeit als Dielektrikum bzw. Besohlung untersucht wurde. (Mitte) Verklebung der untersten drei Fußschichten. (rechts) Die fertig verklebten obersten beiden Schichten sind mit der Befestigungsschraube verbunden.*

gende Leuchtdioden aufgelötet. Letztere dienen als Signalgeber und können das Ausgangssignal eines Neurons anzeigen.

Vier speziell geätzte Kupferflächen auf der Platinenunterseite bilden zusammen mit einer vollflächigen Kupferkreisscheibe einen Spezialkondensator. Wie auf dem Ersatzschaltbild sichtbar, handelt es sich um vier zusammengeschaltete variable Kondensatoren, die mechanisch miteinander gekoppelt sind. Als Dielektrikum kommt extrem weicher Silikonkautschuk mit nur fünf Shore-Graden (Shore-A) zum Einsatz.

Je nach Richtung und Stärke des am Fuß wirkenden Kraftvektors werden die Kondensatoren C_i, C_v, C_h und C_{ges} unterschiedlich stark gequetscht und nehmen dabei Werte von 2–12pF an. Die zu C_i gehörige Kupferfläche C ist hierbei stets nach innen ausgerichtet, zeigt also zum Rumpf hin. C_v und C_h werden beim Anstecken des Beins je nach Körperseite entsprechend definiert. Die Kondensatorwerte werden ständig ausgewertet und dem System als neuronale Sensoraktivitäten zur Verfügung gestellt.

Die Entwicklung des Fußsensors machte intensive Materialrecherchen und -tests notwendig; eine vorläufige Entscheidung fiel zugunsten des entsprechenden Silikonkautschuks für das Dielektrikum und des Moosgummis mit maximaler Haftreibung für die Besohlung (siehe Abbildung 11.10).

Der fertig aufgebaute Fuß stellt ein in sich geschlossenes, unabhängiges Sensorsystem dar. Es besitzt eine nachführende Selbstkalibrierung und die vollständig gekapselte SMD-Elektronik ist vor Berührung geschützt und störungssicher gegenüber der Einstreuung von Netzbrummen oder Hochfrequenzsignalen. Die zuverlässigen Sensordaten und der einfache Anschluss machen es auch für andere Anwendungen interessant.

11.5 Humanoide Roboter

Das folgende Kapitel beschreibt die als A-Serie bezeichneten humanoiden Roboter des Humanoid Team Humboldt, die für Forschung und Lehre sowie im Rahmen der jährlich stattfindenden RoboCup-Wettkämpfe zum Einsatz kommen. Abbildung 11.11 zeigt einen der fünf Roboter. Das Humanoid Team existiert seit Januar 2006, doch die Entwicklung der A-Serie begann bereits Ende 2005 und die verwendeten Konzepte und elektronischen Schaltungen basieren zum großen Teil auf den anderen hier vorgestellten Roboterplattformen.

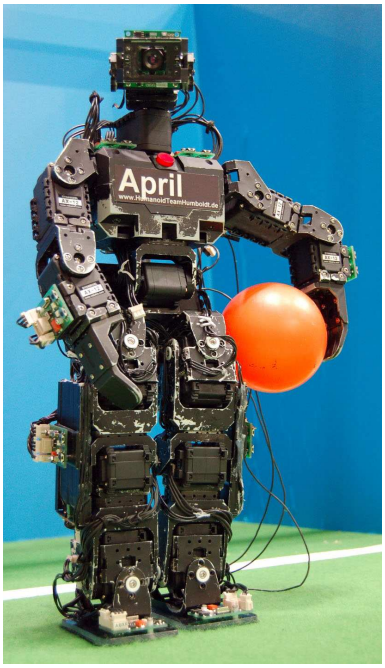

Abbildung 11.11: *Roboter* April *vom Humanoid Team Humboldt.*

Mechanisches Design

Alle fünf Roboter des Humanoid Teams basieren auf dem aus Korea stammenden *Bioloid Robot Construction Kit* der Firma Robotis. Aus diesem Baukastensystem werden etliche Konstruktionsteile aus Plastik sowie die sensomotorischen Aktuatoren (Dynamixel AX-12 / AX-12+) verwendet.

Zusätzlich kommen sogenannte *AccelBoards* zum Einsatz, die speziell für die A-Serie entwickelt wurden und auf verteilte Weise sensomotorische Daten verarbeiten können. Die AccelBoards sind an den verschiedenen Körperex-

tremitäten des Roboters befestigt, so dass die unterschiedlichen Körperregionen sensorisch erfasst werden können. Die Roboter sind auch mit einer nach oben, unten, links und rechts schwenkbaren Kamera ausgestattet, wie in Abbildung 11.12 auf der nächsten Seite zu sehen ist.

Der untere Torso enthält zwei Lithium-Polymer-Batterien in den Oberschenkeln und eine dritte im oberen Torso. Dort ist ebenfalls ein PDA (Pocket Loox, Fujitsu-Siemens) angebracht, der als visuelles Subsystem fungiert.

Jeder Roboter besitzt 21 Freiheitsgrade: sechs pro Bein, drei pro Arm, einen zwischen Hüfte und oberem Torso sowie zwei für die Kamerabewegungen. Die Gesamtgröße beträgt 42cm und das Gesamtgewicht 2.1kg. Die Betriebsspannungsversorgung erfolgt über die drei Lithium-Polymer-Batterien (je 2450mAh), außer beim PDA – dieser besitzt einen eigenen Akku. Die Spannungsversorgung wird über ein Bussystem an alle Aktuatoren und Submodule verteilt.

Aktuatoren

Alle Gelenke werden von Dynamixel AX-12 oder AX-12+ Aktuatoren angetrieben. Es handelt sich hierbei um intelligente, modulare Aktuatoren, welche ein Getriebe, einen Gleichstrommotor, einen Winkelsensor und einen Mikrocontroller mit serieller Schnittstelle in einem Gehäuse beinhalten. Trotz der kompakten Baugröße können die Aktuatoren ein hohes Drehmoment produzieren und widerstehen großen externen Kräften. Die wichtigsten Parameter lauten (bei 10V Betriebsspannung):

- Masse: 55g
- Übersetzungsverhältnis des Getriebes: 1/254
- Maximales Haltedrehmoment: 165 Ncm
- Geschwindigkeit: 0.196s/60°
- Auflösung: 0.35°

Im Vergleich zu Standard-Servos besitzen die eingesetzten Aktuatoren ein Hochgeschwindigkeits-Bussystem und werden mit einer vergleichsweise hohen Spannung betrieben (bis zu 13V). Außerdem können über den Datenbus sensorische Werte ausgelesen werden.

Sensoren

Es existieren die folgenden drei sensorischen Subsysteme:

- Kamera mit PDA zur Bildverarbeitung
- Sensorische Signal der Aktuatoren
- Beschleunigungsdaten

Das Kamerasystem besteht aus einem preiswerten Farbkamera-Modul, dessen Videosignal von einem sogenannten *Framegrabber* digitalisiert wird. Dieser steckt im CompactFlash-Anschluss des PDAs. Das Kamerasystem ist in Abbildung 11.12 als Detailaufnahme zu sehen.

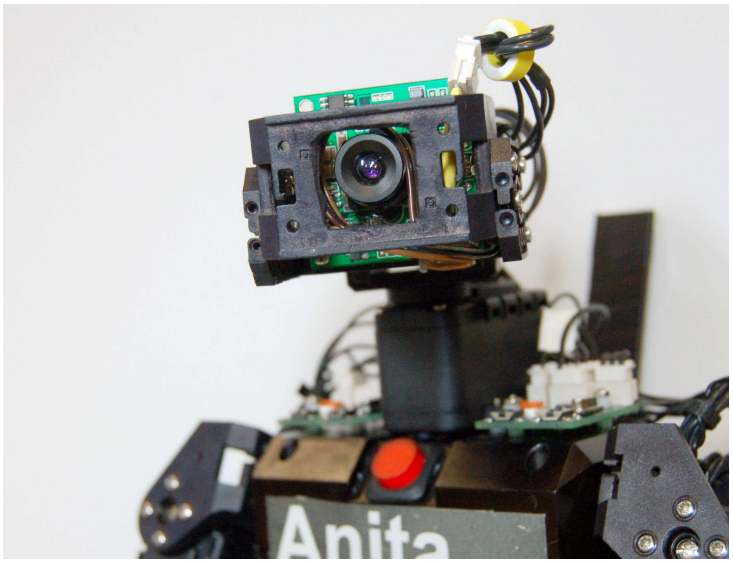

Abbildung 11.12: *Die Roboter besitzen eine schwenkbare Farbkamera.*

Die von den Aktuatoren stammenden Sensorsignale beinhalten die jeweils aktuelle Winkelposition des Gelenks, die aktuelle Motorgeschwindigkeit sowie diagnostische Daten (Temperatur, Betriebsspannungsüberwachung).

Zusätzliche Beschleunigungswerte stehen über die selbst entwickelten AccelBoards zur Verfügung, welche ein aufbereitetes Signal der Zwei-Achsen-Beschleunigungssensoren ADXL213 (Analog Devices) über den Bus senden. Wie in Abbildung 11.13 gezeigt, lassen sich die AccelBoards auf einfache Weise in allen Orientierungen an beliebigen Körperpositionen befestigen.

Insgesamt werden pro Roboter acht AccelBoards verwendet, davon befinden sich zwei auf den Schultern, zwei auf den Oberarmen, zwei in Kniehöhe

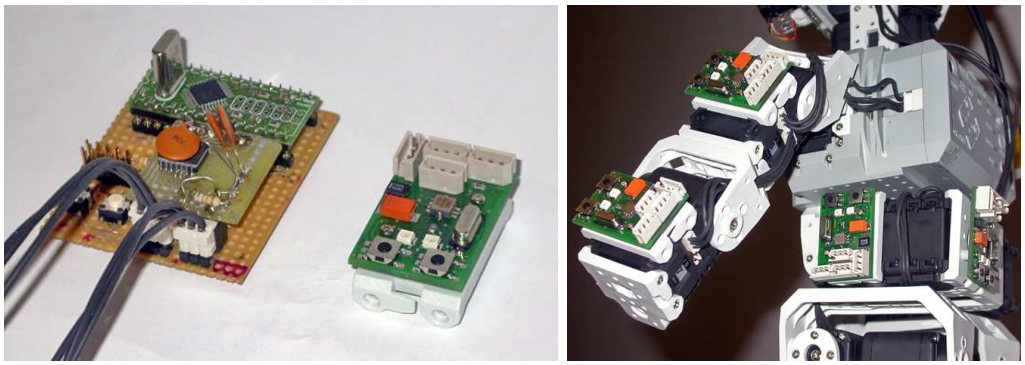

Abbildung 11.13: *(links) Prototyp und fertiges AccelBoard in SMD-Technik im Vergleich. (rechts) Die AccelBoards lassen sich an vielen Körperstellen montieren.*

und schließlich zwei auf den Füßen (jeweils eins auf der linken und eins auf der rechten Körperseite). Mit Hilfe der Beschleunigungsdaten lassen sich unterschiedliche Roboterposen voneinander unterscheiden, was in Kombination mit den Winkeldaten der Aktuatoren umfassenden Aufschluß über den Körperzustand des Roboters liefert. Insbesondere ist auf diese Weise leicht zu detektieren, ob und auf welche Seite ein Roboter umgefallen ist, so dass mit der entsprechenden Aufstehbewegung begonnen werden kann. Die AccelBoards sind nahtlos in das Bussystem der Aktuatoren integriert.

Prozessoren

Auf den Robotern arbeiten zwei Prozessorsysteme parallel. Der PDA verwendet einen PXA272 (Intel) mit einer Taktrate von 520MHz und führt Bildverarbeitung, Umweltmodellierung und Verhaltenssteuerung aus. Zusätzlich besitzt jedes der acht AccelBoards freie Rechenleistung, die zur Berechnung der Bewegungsdaten eingesetzt wird. Die AccelBoards verwenden RISC Mikrocontroller von Renesas.

Inter-Prozessor-Kommunikation

Der PDA, die Dynamixel Aktuatoren sowie die AccelBoards sind über ein Bussystem miteinander verbunden. Der PDA und der Master der AccelBoards kommunizieren über eine bidirektionale serielle Schnittstelle (RS-232, full duplex). Die Kommunikation zwischen Master und den restlichen AccelBoards sowie den Dynamixel Aktuatoren erfolgt mit hoher Geschwindigkeit über RS-485 (1MBaud, half duplex).

Alle AccelBoards operieren auf einem gemeinsamen Bus, über welchen sensomotorische Daten ausgetauscht werden, daher wird dieser Bus als *Spi-*

nalCord (Rückenmark) bezeichnet. Zusätzlich sind an jedes AccelBoard über eine zweite, dem SpinalCord abgewandte Schnittstelle alle nahe gelegenen Aktuatoren angeschlossen. Die AccelBoards berechnen die notwendigen Motorsignale und übertragen diese direkt zu den Aktuatoren. Alle AccelBoards senden sämtliche lokal erfassten sensorischen Signale und alle lokal berechneten Daten auf den SpinalCord, so dass diese prinzipiell auf jedem Prozessor zur Verfügung stehen. Der PDA kann daher als übergeordnetes System beliebige Informationen vom sensomotorischen System anfordern.

Bewegungsansteuerung und Simulationsumgebung

Die motorische Ansteuerung erfolgt, wie in Kapitel 6.1 beschrieben, über die sogenannte Keyframe-Methode. Da die Aktuatoren in einen widerstandsfreien Zustand versetzt werden können und gleichzeitig die aktuellen Winkelwerte auslesbar bleiben, lässt sich der Roboter manuell verformen und die so gewonnenen Posen können als Keyframe abgespeichert werden. Hintereinander abgespielte Keyframes bringen den Roboter in die gewünschte Bewegung.

Zusätzlich existiert eine Simulationsumgebung (siehe Abbildung 6.9 auf Seite 76), die sowohl Keyframe-basierte Daten lesen und schreiben kann, als auch neuronale Netze als Ansteuerungsmethode zur Verfügung stellt. Insbesondere lassen sich in der Simulationsumgebung neuronale Netze evolvieren (siehe [Hei07]), die später auf der realen Hardware probiert werden können.

Kapitel 12

Zusammenfassung des praktischen Teils

Betrachtet man nochmals den in der Einleitung des praktischen Teils vorgestellten Stammbaum (siehe Abbildung 8.1 auf Seite 84), dann sieht man, dass die in den letzten Kapiteln vorgestellen Systeme im zeitlichen Verlauf einem inhaltlichen Plan gefolgt sind.

Am Anfang stand die Erfahrung mit der Fahrenden Platine. Im Laufe der nächsten zwei Jahre weiteten sich die untersuchten Fragestellungen in die Breite aus und die Anzahl der entwickelten Systeme erhöhte sich allmählich. Dabei wurden einerseits vorhandene Designs weiterentwickelt, andererseits kamen neue Fragestellungen hinzu, die zunächst mit Hilfe spezieller Versuchsaufbauten isoliert untersucht wurden. Im Laufe der darauf folgenden Jahre wurden die gewonnenen Erkenntnisse auf den beiden Robotern Do:Little und Oktavio vereint und im letzten Jahr wurde zusätzlich die humanoide A-Serie fertiggestellt, so dass sich der Stammbaum zum Schluss hin wieder verjüngt.

Das Gesamtvorhaben, neuronale Netze zur Bewegungssteuerung, und etwas weitergefasst, im Hinblick auf Sensomotorik zu untersuchen, wurde zur inhaltlichen Bearbeitung in drei gleichberechtigte Bereiche aufgeteilt:

1. Experimentalsysteme zur Untersuchung einer kognitiven Einzelleistung (Bewegungswahrnehmung, Phonotaxis)

2. Autonome, mobile Verbundsysteme mit Schwerpunkt Sensorik (Fahrende Platine, Do:Little)

3. Autonome, mobile Verbundsysteme mit Schwerpunkt Motorik (Lucy, TED, Oktavio)

Alle aufgebauten Systeme dienten der Überprüfung theoretischer Konzepte, lieferten aber stets auch Impulse zurück in die Theorie – sowohl was die Gewichtsparameter und Konnektivität neuronaler Netze angeht, als auch die Wechselwirkung von Morphologie und neuronaler Steuerung betreffend.

Designprinzipien

Bei der Entwicklung der beschriebenen Systeme wurden vier Kriterienbündel berücksichtigt, die im folgenden besprochen werden. Das erste und herausforderndste Kriterienbündel lautet:

- Kostengünstig
- Schnell aufzubauen
- Einfach und minimalistisch

Obwohl man zunächst den Eindruck bekommen kann, es würde sich um die dreimalige Wiederholung derselben Forderung handeln, ist leider das Gegenteil der Fall. Oft gibt es ein teures Spezial-IC, das die benötigte Funktionalität in einem 8-beinigen Gehäuse ohne externe Bauteile bietet. Die Schaltung ist dann natürlich schnell aufgebaut – nur in der Regel erheblich kostenintensiver, als die Variante mit Standardbauteilen. Hier gilt es also abzuwägen, was mit welchem Aufwand vertretbar ist und wie der vorher fixierte Kostenrahmen am besten verteilt wird. Oft lässt sich eine Lösung finden, die das gesteckte Ziel über einen neuen, ungewöhnlichen Weg erreicht, wie zum Beispiel die Extraktion der Vertikalsynchronisation bei der Bewegungswahrnehmung (Kapitel 9.3), die aktiven Gradientensensoren am Do:Little (Kapitel 10.2) und der Fußsensor des Oktavios (Kapitel 11.4).

Die Forderung nach einem minimalistischen System hat neben dem Aspekt des Schaltungsdesigns noch eine substanziellere Dimension. Diese setzt bereits bei der exakten Formulierung des Ziels ein. Fordert man explizit eine vierbeinige Laufmaschine, oder will man eigentlich nur einen Laufroboter, der nach vorne, links und rechts gehen kann? Im zweiten Fall endet man bei konsequenter Berücksichtigung der genannten Kriterien beim Krabbelroboter (Kapitel 11.3).

Das zweite Kriterienbündel ist in sich stimmiger:

- Modular
- Dezentral
- Kombinierbar

Es handelt sich zwar wieder um drei unterschiedliche Kriterien, doch sind diese einfacher gleichzeitig zu erfüllen. Bei den ersten Systemen bedeutete die Berücksichtigung noch einen Mehraufwand, aber danach machte sich die Wiederverwertbarkeit einzelner Module positiv bemerkbar. Anderenfalls hätte der Do:Little nicht in fünf Monaten bis zur Kleinserie reifen können. Das augenfälligste Beispiel für umgesetzte Modularität ist sicherlich das funktional gekapselte Einzelbein des Oktavios. Darüberhinaus ist hier die Wiederverwendung von Schaltungsmodulen und Funktionsblöcken zu nennen.

Die Forderung nach einem dezentralen Aufbau kann auf zweierlei Weise verstanden werden. Erstens wie im Fall des Oktavio, wo jedes Bein mit einem eigenen Mikrocontroller ausgestattet ist und das Gesamtsystem als solches keine zentrale Rechenleistung besitzt. Zweitens wie bei der Bewegungswahrnehmung, wo eine kleine elektronische Schaltung den Mikrocontroller deutlich entlastet und somit im Hinblick auf die Rechenleistung eine dezentralisierende Funktion hat. Ohne diese Schaltung wäre der Mikrocontroller gezwungen, das Videosignal periodisch mit hoher Frequenz einzulesen, um den Zeitpunkt der Vertikalsynchronisation präzise bestimmen zu können.

Das dritte Kriterienbündel zwingt den Entwickler zur Fleißarbeit:

- Autonom und effizient
- Leicht und mobil
- Robust und störsicher

Die einzelnen Kriterien sind meist irgendwie erreichbar, sie setzen nur entsprechende Materialrecherchen und die richtige Schaltungsidee voraus. Das wesentliche ist eine ausgefeilte Regelung des Energiehaushalts, wie beim Do:Little gezeigt wurde. Sein Energievorrat reicht für mehrere Stunden, er kann ihn aus einer Quelle von 1–13V Spannung wieder aufladen und sogar eigene Energie an andere Do:Littles abgeben. Ähnliche Konzepte findet man auch beim Oktavio. In der Vorspannfeder seiner Beine zwischengespeicherte oder extern zugeführte kinetische Energie wandelt er mit Hilfe seiner Motoren in elektrische um und führt diese den Akkus zu.

Zur Energieeffizienz und Mobilität trägt ein geringes Gewicht wesentlich bei. Neben TED ($m = 90$g) und Lucy ($m = 0.3$kg) kann auch Oktavio mit seiner Gesamtmasse von knapp 14kg als leicht bezeichnet werden, wenn man die Größenverhältnisse entsprechend berücksichtigt.

Zum Thema Robustheit und Störsicherheit zum Beispiel die vom Umgebungslicht unabhängigen Gradientensensoren des Do:Little sowie dessen extrem störsichere akustische Kommunikationsfähigkeit hervorzuheben.

Das letzte Kriterienpaar ist eher von philosophischer Natur:

- Authentisch
- Erfahrbar

Die geforderte Authentizität bezieht sich darauf, dass sämtliche Sensorqualitäten, die dem neuronalen Netz zur Verfügung gestellt werden, nicht in irgendeiner Weise emuliert oder simuliert werden, sondern der physikalischen Realität des Systems entsprechen.

Hierzu ein Beispiel: Häufig wird bei Experimenten zum Energiehaushalt bei Robotern eine interne Variable eingeführt, welche die zur Verfügung stehende Energie repräsentiert und langsam auf Null heruntergezählt wird. Die Variable muss natürlich schneller bei Null ankommen als der reale Akkustand, da die Emulation ansonsten vorher zusammenbricht.

Bei den hier vorgestellten Systemen wurden solche Verfahren strikt abgelehnt – das entsprechende Eingangsneuron teilt immer den realen Energiezustand mit. Erstens nutzt man die Laufzeit des Akkus auf diese Weise voll aus, zweitens wird dem Roboter bei der authentischen Schaltungsvariante auch erfahrbar gemacht, dass sich sein motorisches Verhalten unmittelbar auf den Energievorrat auswirkt.

Sind alle bisherigen Kriterien erfüllt, dann bleibt zuletzt die Frage, ob das für den Aussenstehenden beobachtbare Verhalten des Roboters ausreichend gut mit menschlichen Sinnen erfahrbar ist. Diese Forderung verbietet somit alle Funkmodule wie Bluetooth oder WLAN. Letztendlich war dies mit dafür ausschlaggebend, dass die Do:Littles untereinander über akustische Signale kommunizieren.

Zentrale Ergebnisse

Durch die Entwicklung und den Aufbau der Robotiksysteme sowie die damit durchgeführten Experimente konnten die theoretischen Konzepte untermauert werden. Darüber hinaus haben sich aber auch Erfahrungen angesammelt, die direkt beim Aufbau der Systeme gemacht wurden und nur schwer aus rein theoretischen Überlegungen entstanden wären. Diese Erfahrungen lassen sich als zentrale Ergebnisse des praktischen Teils festhalten:

1. Die Morphologie des Systems muss gut durchdacht sein, die Verwendung von speziell angefertigten Teilen ist dabei manchmal nicht zu umgehen (Teile aus Kohlefaserverbundstoff beim Oktavio, speziell gebogene Kontaktfedern beim Do:Little, menschliche Ohrnachbildung bei der Phonotaxis).
2. Intensive Materialrecherchen inklusive Tests und Selektion benötigen viel Zeit und müssen von Anfang an mit eingeplant werden (elastische

Beine bei Lucy, Fußbeschaffenheit beim Krabbelroboter, Signalgeber, Mikrofone, Servos und mehr beim Do:Little, Dielektrikum des Fußsensors beim Oktavio, um nur einige Beispiele zu nennen).

3. Lithium-Polymer-Akkus sind trotz aufwendiger Laderegelung zur Zeit der optimale Energiespeicher für kleine bis mittlere Robotiksysteme (Do:Little, Oktavio, Humanoide A-Serie).

4. Der verwendete Mikrocontroller sollte ein schnelles, reaktives System ermöglichen, aber nicht unnötig groß gewählt werden. Man kann leider die allgemeine Tendenz beobachten, dass Laptops, PDAs und andere stromfressende High-Performance-Mikrocontroller verwendet werden, wo ein einfacher Mikrocontroller gereicht hätte. Das Gegenbeispiel par excellance ist hier der TED, dessen Prozessor bei nur 1.6MHz und ohne Arbeitsspeicher trotzdem sechs vollständig verknüpfte Neuronen in Echtzeit berechnen kann.

5. Prinzipiell sollte die sensomotorische Schleife stets vollständig durch das neuronale Netz geleitet werden. Servos verletzen dieses Prinzip bereits (siehe Experiment mit dem Universalgreifer). Verzichtet man auf Servos, fallen als zusätzlicher positiver Nebeneffekt auch die Shuntfilter weg (Oktavio).

6. An propriozeptiven Sensoren sollte nicht gespart werden. Je mehr dem System zur Verfügung stehen, desto vielfältiger sind die semantischen Interpretationsmöglichkeiten, welche durch die Verknüpfung der Signale entstehen. Man könnte sagen, die Eigenwahrnehmung des Systems erreicht dadurch eine höhere Stufe. Es macht einen Unterschied, ob bei sinkendem Energiepegel nur die subjektive Empfindsamkeit des Roboters steigt, oder ob dieser über ein propriozeptives Energieneuron auch die Information hat, dass seine Energie gerade zur Neige geht und er daher empfindlicher als sonst wahrnimmt. Wagt man sich sehr weit vor, könnte man dies als notwendige Grundlage für eine Form der Selbsterkenntnis bezeichnen.

7. Real aufgebaute Systeme können emergent oder zumindest nicht vorprogrammiert biologische Verhaltensweisen zeigen. Beispiel hierfür ist das künstliche Neuron, welches bei sinkender Energieversorgung mehr sensorische Stimulation benötigt um zum gleichen Aktivierungszustand zu kommen und die Wahrnehmung externer Stressoren bei der Fahrenden Platine, welche dieselbe Hürde bei sinkender Batteriespannung als schwerer wahrnimmt.

Fazit

Wie man sieht, greifen in der Robotik Theorie und Praxis auf besondere Weise ineinander, so dass rückblickend bestätigt wird, dass es keinen Sinn macht, das eine ohne das andere zu betrachten. Interessant ist hierbei die Tatsache, dass manche Systeme vom biologischen Vorbild inspiriert sind, andere hingegen biologische Verhaltensweisen zeigen, ohne dass dies vorher konzeptionell vorgesehen wurde.

Mit den Systemen Do:Little, Oktavio und der Humanoiden A-Serie sind trotz minimalem Kostenrahmen drei Roboterplattformen mit interessanten Alleinstellungsmerkmalen entstanden. Neben diesen eignen sich aber auch viele andere der vorgestellten Systeme als experimentelle Plattform für universitäre und außeruniversitäre Projektarbeiten. So wurden neben dem beschriebenen Gruppenprojekt mit TED, auch die Systeme Bewegungswahrnehmung, Phonotaxis und Fahrende Platine bereits im Rahmen von Projektkursen, Studien- und Diplomarbeiten mit Studenten erfolgreich eingesetzt.

Kapitel 13

Ausblick

Die vorgestellten neurodynamischen Module zur Bewegungssteuerung autonomer Roboter wurden analysiert, simuliert und wo möglich auch auf existierenden Roboterplattformen zu Demonstrationszwecken implementiert. Was die monostabilen Neuromodule angeht, steht jedoch eine umfangreiche Testphase im Zusammenspiel mit parallel operierenden Systemkomponenten noch aus. Hierzu müssen etliche Softwarekomponenten angepasst werden, Messreihen aufgenommen und mit Resultaten aus der Simulationsumgebung verglichen werden.

Im Rahmen betreuter Studien- und Diplomarbeiten soll die Simulationsumgebung nach und nach den physikalischen Eigenheiten der verwendeten humanoiden Roboter (A-Serie) angepasst werden, so dass die in Kapitel 6 angesprochenen Evolutionen zur Bewegungsstabilisierung durchgeführt werden können. Neben den sensorischen Winkel- und Beschleunigungsdaten (siehe Kapitel 11.5) ist eine Exploration des optischen Flusses im Hinblick auf sensomotorische Einkopplung vielversprechend. Auch hier haben die Arbeiten bereits begonnen; erste Resultate liegen in Form einer Diplomarbeit vor (siehe [Wol07]).

Was die theoretische Fortführung der Neuromodule betrifft, so ist es naheliegend, die Sammlung an neurodynamischen Modulen kontinuierlich zu erweitern. Konzepte verschiedener Wahrnehmungsmodule existieren schon für visuelle und auditive Signale. Im praktischen Teil der vorliegenden Arbeit wurde bereits auf die entsprechenden Versuchsaufbauten eingegangen. Außerdem sind Ansätze für die langfristige Speicherung kontinuierlicher Signale vorhanden. Im Hinblick auf die Evolution umfassender neuronaler Netze, die von der sensorischen Verarbeitung bis hin zur motorischen Steuerung alles beinhalten, wäre es von Vorteil den Evolutionsalgorithmus mit Operatoren auszustatten, die anstelle einzelner Neuronen auch Neuromodule oder noch größere Strukturen als Entität einsetzen und verknüpfen können.

Bezüglich der Hardware bleibt – wie in den vergangenen Jahren – die Aufgabe bestehen, kontinuierlich den sich schnell ändernden Markt von mechanischen und elektronischen Komponenten zu beobachten, neue Entwicklungen in den Bereichen Sensorik, Motorik und Prozessortechnologie prototypisch auszutesten und gegebenenfalls einzusetzen. Die Erfahrungen mit den im praktischen Teil vorgestellten Systemen werden bei der Konzeption einer neuen humanoiden Baureihe (M-Serie) mit einfließen.

Literaturverzeichnis

[Ame03] Ames, J. C.: *Design Methods For Pattern Generation Circuits*, Department of Electrical Engineering and Computer Science, Case Western Reserve University, Diplomarbeit, 2003

[AZR04] Afraimovich, V. S. ; Zhigulin, V. P. ; Rabinovich, M. I.: On the Origin of Reproducible Sequential Activity in Neural Circuits. In: *Chaos* 14 (2004), Nr. 4, S. 1123–1129

[Bac05] Bachmann, F. *Onboard-Evolution (Arbeitstitel)*. Studienarbeit am Lehrstuhl für Künstliche Intelligenz, Humboldt-Universität zu Berlin. 2005

[Bar93] Barron, A. R.: Universal Approximation Bounds for Superpositions of a Sigmoidal Function. In: *IEEE Transactions on Information Theory* 39 (1993), S. 930–945

[Bau05] Bauer, R. A. *Phonotaxis – Richtungslokation mittels genetischer Evolution*. Studienarbeit am Lehrstuhl für Künstliche Intelligenz, Humboldt-Universität zu Berlin. 2005

[Bea79a] Beauchamp, J.: Brass Tone Synthesis by Spectrum Evolution Matching with Nonlinear Functions. In: *Computer Music Journal* 3 (1979), Nr. 2, S. 35–43

[Bea79b] Beauchamp, J.: Practical Sound Synthesis Using a Nonlinear Processor (Waveshaper) and a High-Pass Filter. In: *Computer Music Journal* 3 (1979), Nr. 3, S. 42–49

[Bee93] Beebe, N. H. F.: Accurate Hyperbolic Tangent Computation. In: *Notes for Mathematics* 119 (1993)

[BF02] Blynel, J. ; Floreano, D.: Levels of Dynamics and Adaptive Behavior in Evolutionary Neural Controllers. In: *Proceedings of the International Conference on the Simulation of Adaptive Behavior* (2002)

[BgD+99] Bayraktaroglu, I. ; Ögrenci, A. S. ; Dündar, G. ; Balkir, S. ; Alpaydin, E.: ANNSyS: an Analog Neural Network Synthesis System. In: *Neural Networks* 12 (1999), S. 325–338

[BPVL94] Beiu, V. ; Peperstraete, J. A. ; Vandewalle, J. ; Lauwereins, R.: VLSI Complexity Reduction by Piece-Wise Approximation of the Sigmoid Function. In: Verleysen, M. (Hrsg.): *Proceedings of the European Symposium on Artificial Neural Networks*, 1994, S. 181–186

[BS03] Babitsky, V. I. ; Shipilov, A. V.: *Resonant Robotic Systems*. Springer Verlag, 2003

[CBD+97] Canavier, C. C. ; Butera, R. J. ; Dror, R. O. ; Baxter, D. A. ; Clark, J. W. ; H., Byrne J.: Phase Response Characteristics of Model Neurons Determine Which Patterns Are Expressed in a Ring Circuit Model Of Gait Generation. In: *Biological Cybernetics* 77 (1997), S. 367–380

[Cyb89] Cybenko, G.: Approximation by Superposition of a Sigmoidal Function. In: *Mathematics of Control, Signal and Systems* 2 (1989), S. 303–314

[DMPS01] Daucé, E. ; Moynot, O. ; Pinaud, O. ; Samuelides, M.: Mean-field Theory and Synchronization in Random Recurrent Neural Networks. In: *Neural Processing Letters* 14 (2001), S. 115–126

[DP99] Der, R. ; Pantzer, T.: Emergent Robot Behavior From the Principle of Homeokinesis. In: *Proceedings of the 1st International Khepera Workshop* (1999)

[DP02] Di Paolo, E. A.: Fast Homeostatic Neural Oscillators Induce Radical Robustness In Robot Performance. In: *Proc. of 7th Int. Conference on Simulation of Adaptive Behavior* (2002), S. 303–304

[FGA00] Fusi, S. ; Giudice, P. D. ; Amit, D. J.: Neurophysiology of a VLSI spiking neural network: LANN21. In: *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks*, 2000

[GCA] Goldberg, D. H. ; Cauwenberghs, G. ; Andreou, A. G. *Analog VLSI Spiking Neural Network With Address Domain Probabilistic Synapses*

[Gir] Girau, B. *Digital hardware implementation of 2D compatile neural networks*

[GK02] Gerstner, W. ; Kistler, W. M.: *Spiking Neuron Models.* Cambridge University Press, 2002

[GPS03] Golubitsky, M. ; Pivato, M. ; Stewart, I.: Interior Symmetry and Local Bifurcation in Coupled Cell Networks. In: *Dynamical Systems: an International Journal* 19 (2003), S. 389–407

[GS85] Gordon, J. W. ; Smith, J. O.: A Sine Generation Algorithm for VLSI Applications. In: *Proc. of Int. Computer Music Conf.* (1985), S. 165–168

[GS93] Golubitsky, M. ; Stewart, I.: An Algebraic Criterion for Symmetric Hopf Bifurcation. In: *Proceedings: Mathematical and Physical Sciences* 440 (1993), S. 727–732

[GSBC98] Golubitsky, M. ; Stewart, I. ; Buono, P.-L. ; Collins, J. J.: A Modular Network for Legged Locomotion. In: *Physica D* 105 (1998), S. 56–72

[GST05] Golubitsky, M. ; Stewart, I. ; Torok, A.: Patterns of Synchrony in Coupled Cell Networks with Multiple Arrows. In: *SIAM Journal of Applied Dynamical Systems* (to appear in 2005)

[Ham] Hammer, B. *Approximation Capabilities of Folding Networks*

[Hei07] Hein, D.: *Simloid: Evolution of Biped Walking Using Physical Simulation: Diplomarbeit*, Institut für Informatik, Humboldt Universität zu Berlin, Diplomarbeit, 2007

[Hig90] Higgins, R. J.: *Digital Signal Processing in VLSI.* Prentice-Hall, 1990. – 524–555 S

[Hik03] Hikawa, H.: A Digital Hardware Pulse-Mode Neuron With Piecewise Linear Activation Function. In: *IEEE Transactions on Neural Networks* 14 (2003), Nr. 5, S. 1028–1037

[HNS92] Harrer, H. ; Nossek, J. A. ; Stelzl, R.: An Analog Implementation of Discrete-Time Cellular Neural Networks. In: *IEEE Transactions on Neural Networks* 3 (1992), Nr. 3, S. 466–476

[HP07] Hild, M. ; Pasemann, F.: Self-Adjusting Ring Modules (SARMs) for Flexible Gait Pattern Generation. In: *Proc. of Int. Joint Conf. on Artificial Intelligence (IJCAI)* (2007)

[HSW89] Hornik, K. ; Stinchcombe, M. ; White, H.: Multilayer Feedforward Networks are Universal Approximators. In: *Neural Networks* 2 (1989), S. 359–366

[ITN03] Inamura, T. ; Tanie, H. ; Nakamura, Y.: Keyframe Compression and Decompression for Time Series Data Based on the Continuous Hidden Markov Model. In: *Proc. of Int. Conf. on Intelligent Robots and Systems (IROS)* 2 (2003), S. 1487–1492

[Jon90] Jones, L. K.: Constructive Approximations for Neural Networks by Sigmoidal Function. In: *IEEE Proceedings* 78 (1990)

[KB03] Küfner, P. ; Buhle, S. *Populationsmanager: Programm zur künstlichen Evolution neuronaler Netze.* Seminararbeit Evolution und Robotik, Humboldt-Universität zu Berlin (LFG KI). 2003

[KK98] Kitajima, H. ; Kawakami, H.: Bifurcation of Periodic States in Coupled BVP Oscillators With Hard Characteristics. In: *Proceedings of NOLTA98* (1998), S. 381–384

[Lai98] Laing, C. R. *Rotating Waves in Rings of Coupled Oscillators.* 1998

[MH03] Milev, M. ; Hristov, M.: Analog Implementation of ANN With Inherent Quadratic Nonlinearity of the Synapses. In: *IEEE Transactions on Neural Networks* 14 (2003), Nr. 5, S. 1187–1200

[MM92] Mhaskar, H. N. ; Micchelli, C. A.: Approximation by Superposition of Sigmoidal and Radial Basis Functions. In: *Advances in Applied Mathematics* 13 (1992), S. 350–373

[MP43] McCulloch, W. ; Pitts, W.: A logical calculus of the ideas immanent in nervous activity. In: *Bulletin of Mathematical Biophysics* 5 (1943), S. 115–133

[MRTAE00] Maya, S. ; Reynoso, R. ; Torres, C. ; Arias-Estrada, M.: Compact Spiking Neural Network Implementation in FPGA. In: Hartenstein, R. W. (Hrsg.) ; Grünbacher, H. (Hrsg.): *FPL 2000, LNCS 1896*, Springer-Verlag, 2000, S. 270–276

[MT00] Maeda, Y. ; Tada, T.: FPGA Implementation of a Pulse Density Neural Network Using Simultaneous Perturbation. In: *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks*, 2000

[NAA03] Nakada, K. ; Asai, T. ; Amemiya, Y.: An Analog Neural Oscillator Circuit for Locomotion Controller in Quadruped Walking Robot. In: *Proceedings of the International Joint Conference on Neural Networks* 2 (2003), S. 983–988

[NF00] Nolfi, S. ; Floreano, D.: *Evolutionary Robotics*. MIT Press, 2000

[NP99] Nolfi, S. ; Parisi, D.: Exploiting the Power of Sensory-Motor Coordination. In: *Advances in Artificial Life: Proceedings of the 5th European Conference (ECAL '99)* (1999)

[NT04] Nishimoto, R. ; Tani, J.: Learning to Generate Combinatorial Action Sequences Utilizing the Initial Sensitivity of Deterministic Dynamical Systems. In: *Neural Networks* 17 (2004), Nr. 7, S. 925–933

[Pas95] Pasemann, F.: Characterization of Periodic Attractors in Neural Ring Networks. In: *Neural Networks* 8 (1995), S. 421–429

[Pas98] Pasemann, F.: Structure and Dynamics of Recurrent Neuromodules. In: *Theory in Biosciences* 117 (1998), S. 1–17

[Pas02] Pasemann, F.: Complex Dynamics and the Structure of Small Neural Networks. In: *Network: Computation in Neural Systems* 13 (2002), S. 195–216

[PG90] Poggio, T. ; Girosi, F.: Networks for Approximation and Learning. In: *IEEE Proceedings* 78 (1990), Nr. 9

[PHZ03] Pasemann, F. ; Hild, M. ; Zahedi, K.: SO(2)-Networks as Neural Oscillators. In: *Proc. of Int. Work-Conf. on Artificial and Natural Neural Networks (IWANN)* (2003), S. 144–151

[Raf04] Raffle, H. S.: *Topobo: A 3-D Constructive Assembly System With Kinetic Memory*, Program in Media Arts and Sciences, MIT, Diplomarbeit, 2004

[RM86] Rumelhart, D. E. ; McClelland, J. L.: *Parallel Distributed Processing.* MIT Press, 1986

[RVSA06] Rabinovich, M. I. ; Varona, P. ; Selverston, A. I. ; Abarbanel, H. D. I.: Dynamical Principles in Neuroscience. In: *Reviews of Modern Physics* 78 (2006), Nr. 4

[SC92] Smith, J. O. ; Cook, P. R.: The Second-Order Digital Waveguide Oscillator. In: *Proc. of Int. Computer Music Conf.* (1992), S. 150–153

[Sch01] Schneider, W.: *Deutsch für Profis: Wege zu gutem Stil.* Goldmann Verlag, 2001

[Str94] Strogatz, S. H.: *Nonlinear Dynamics and Chaos.* Addison-Wesley Publishing Company, 1994

[THGC98] Tiňo, P. ; Horne, B. G. ; Giles, C. L. ; Collingwood, P. C.: Finite State Machines and Recurrent Neural Networks – Automata and Dynamical Systems Approaches. In: *Neural Networks and Pattern Recognition* (1998)

[TS00] Torikai, H. ; Saito, T.: Pulse-coupled Networks of Non-autonomous Integrate-and-Fire Oscillators and Classification Functions. In: *Proceedings of the International Conference on Neural Networks* (2000)

[TS02] Thompson, J. M. T. ; Steward, H. B.: *Nonlinear Dynamics and Chaos.* John Wiley & Sons, Ltd, 2002

[TTO01] Tsujita, K. ; Tsuchiya, K. ; Onat, A.: Decentralized Autonomous Control of a Quadrupedal Locomotion Robot Using Oscillators. In: *Artificial Life Robotics* 5 (2001), S. 152–158

[UF01] Urzelai, J. ; Floreano, D.: Evolution of Adaptive Synapses: Robots with Fast Adaptive Behavior in New Environments. In: *Evolutionary Computation* 9 (2001), Nr. 4, S. 495–524

[Urb03] Urban, A. *Weltsimulator: Programm zur Simulation einer künstlichen Welt in 2D.* Seminararbeit Evolution und Robotik, Humboldt-Universität zu Berlin (LFG KI). 2003

[Wil03] Williamson, M. M.: Oscillators and Crank-Turning: Exploiting Natural Dynamics with a Humanoid Robot Arm. In: *Philosophical Transactions: Mathematical, Physical & Engineering Sciences* 361 (2003), Nr. 1811, S. 2207–2223

[Wol07] Wollstein, A.: *Rekurrente neuronale Netze zur Detektion des optischen Flusses*, Institut für Informatik, Humboldt Universität zu Berlin, Diplomarbeit, 2007

[YNA00] Yoneyama, T. ; Ninomiya, H. ; Asai, H.: Design Method of Limit Cycle Generator By Hysteresis Neural Networks. In: *Proceedings of the International Conference on Neural Networks* (2000)

Danksagung

Mit Freude bedanke ich mich bei denen, die mich bei meiner wissenschaftlichen Arbeit und der Herstellung dieser Dissertationsschrift unterstützt haben. Allen voran danke ich Herrn Prof. Dr. H.-D. Burkhard und Herrn Prof. Dr. F. Pasemann, die mich vollkommen selbstbestimmt arbeiten ließen und stets ein offenes Ohr für meine Fragen und Wünsche hatten.

Ich danke meinen Kollegen im Fraunhofer-Institut AIS für viele fruchtbare Diskussionen, in besonderem Maße den Herren M. Hülse, A. von Twickel, S. Wischmann und K. Zahedi. Beim Aufnehmen von Messreihen und im SMD-Labor haben mich die Herren S. Kubina, P. Manoonpong, M. Rejzek und J. Winzer unterstützt, wofür ich ihnen danke. Die Roboter Do:Little und Oktavio wären ohne die intensive Zusammenarbeit mit Herrn J. Karabasz und Herrn T. Siedel niemals realisierbar gewesen, so dass ich ihnen dafür zu großem Dank verpflichtet bin.

Meinen Kollegen an der Humboldt-Universität zu Berlin habe ich für anregenden fachlichen Austausch zu danken, insbesondere den Herren J. Bach, R. Berger, U. Düffert, D. Göhring, D. Hein, M. Jüngel, M. Lötzsch und M. Spranger. Für die inhaltliche Mitarbeit in meinen Lehrveranstaltungen danke ich allen Teilnehmern, speziell den Herren F. Bachmann, R. Bauer, D. Weese, A. Wollstein und M. Zelke, die teilweise Studien- und Diplomarbeiten bei mir gemacht haben oder anderweitig interessante Resultate produziert haben. Besondere Erwähnung verdient auch das Humanoid Team Humboldt, welches viel Zeit in den Aufbau der humanoiden Roboter investiert hat. Ganz speziell bedanke ich mich bei den Herren R. Meißner, C. Thiele, M. Kubisch, C. Benckendorff, T. Lobig und B. Werner. Frau Prof. Dr.-Ing. B. Meffert und Herrn Dr.-Ing. M. Günther gilt mein Dank für die Bereitstellung von Laborräumen.

Frau Prof. Dr. B. Stemmer und Herr Dr. B. M. Reuter haben mir aufgezeigt, welche Inhalte umzugestalten sind, damit sich fachfremde Leser besser zurechtfinden; auch hierfür vielen Dank. An dieser Stelle möchte ich mich bei Frau A. Schiffel und Herrn A. von Twickel ausgesprochen herzlich dafür bedanken, dass sie diese Dissertationsschrift einer sorgfältigen Durchsicht unterzogen haben. Ihre Anregungen haben es mir ermöglicht, die Qualität der Arbeit anzuheben.

Lebenslauf

Persönliche Daten

Name	Manfred Hild
Geburtstag	4. April 1968

Ausbildung

1974 – 1978	Grundschule am Stephansplatz, Konstanz
1978 – 1987	Alexander-von-Humboldt-Gymnasium, Konstanz
1987 – 1998	Studium Mathematik (Diplom) und Psychologie (Nebenfach) Universität Konstanz
2002 – 2007	Doktorand Informatik (LFG KI), Humboldt-Universität zu Berlin Fraunhofer-Institut AIS, Sankt Augustin

Berufliche Tätigkeiten

1987 – 1988	Werkstudent der Produktentwicklung Systemsoftware CTM Computertechnik Müller GmbH, Konstanz
1989 – 1991	Wissenschaftliche Hilfskraft Informatik, Universität Tübingen Mathematik und Physik, Universität Konstanz
1992 – 1997	Wissenschaftlicher Mitarbeiter und Projektmanager Forschungsinstitut der Kliniken Schmieder, Allensbach
1997 – 1999	Leiter der Produktentwicklung Software Institut für Gesundheits- und Sozialforschung GmbH, Berlin CSG Clinische Studiengesellschaft mbH, Berlin
1999 – 2002	Projektmanager Magic Toons Software GmbH, Potsdam/Babelsberg GFT Technologies AG, Berlin Tchibo Frisch-Röst-Kaffee GmbH, Hamburg

Selbständigkeitserklärung

Ich erkläre hiermit, dass

- ich die vorliegende Dissertationsschrift „Neurodynamische Module zur Bewegungssteuerung autonomer mobiler Roboter“ selbständig und ohne unerlaubte Hilfe angefertigt habe;
- ich mich nicht bereits anderwärts um einen Doktorgrad beworben habe oder einen solchen besitze;
- mir die Promotionsordnung der Mathematisch-Naturwissenschaftlichen Fakultät II der Humboldt-Universität zu Berlin vom 17. Januar 2005 (zuletzt geändert am 13. Februar 2006) bekannt ist.